CRYPTO: YOU’RE DOING IT WRONG

TALES FROM THE TRENCHES

Jon Green – Aruba Security Guy

May 2017
What’s this about?

• Getting a FIPS certificate is (relatively) easy
  – FIPS module developer
  – “FIPS inside”

• Using it correctly…. Not always

• It is possible (and common) for a FIPS-validated IT product to not be using FIPS-validated cryptography
Example 1 – DIY Crypto

• There’s a certified DRBG in this product…

```python
# open the key file for writing
keyfile = open(KEYFILE, 'w')
...
# seed the random number generator
random.seed()
...
# get 128 random bits and write into the file, for AES key
for i in range(16):
    keyfile.write("%c\"%random.randint(0,255))
# get another 128 random bits and write into the file for AES IV
for i in range(16):
    keyfile.write("%c\"%random.randint(0,255))
...
# close the file and return
keyfile.close()
```

• Python’s random.seed() at least tries to use OS entropy sources if available...

• Is this example really a problem?
Example 2 – Forgetting to Use Crypto

• **GMK: Groupwise Master Key**
  - Used in Wi-Fi WPA2 to encrypt broadcast/multicast traffic
  - Should be randomly generated – good news because we have a certified DRBG!

• **Who can spot the flaw?**

```c
/* TODO call this routine to update the GMK */
void
new_gmk()
{
}
```

```c
/* call this routine to update the GMK */
void
new_gmk(struct sap_info *sap)
{
    generate_random(sap->gmk, sizeof (sap->gmk));
    return;
}
```
Example 3 – Large Multi-Module Products

- Is this concerning?

- Product contains compiled C, Java, Python, PHP, Javascript, Go, Bash scripts, ...

- Also includes a FIPS validated crypto library (OpenSSL)

- Is this a problem?
What’s the deal?

• Plain old-fashioned bugs

• Developers don’t really know where crypto is being used
  – Forgotten
  – Third-party / open-source code
  – Multiple frameworks

• Developers know where crypto is being used, but too much work to change…
  – e.g. OpenSSL API is convoluted and poorly documented – can’t figure it out
  – Cross-language APIs are painful and confusing
Finding Implementation Flaws

• FIPS code review!
  - No… this code is outside the core crypto functions

• Common Criteria code review! Bring back EAL4!
  - No… product in example #2 went through EAL4+ with that flaw

• Product testing
  - Maybe… but none of these flaws would be visible in black-box testing

• Security audit code review
  - The option most likely to have success
  - Bug bounty?
### Example 4 – Non-FIPS Ciphers Detected

<table>
<thead>
<tr>
<th>Cipher Suites</th>
<th>#</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc82c) ECDH secp384r1 (eq. 3072 bits RSA) FS</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc32b) ECDH secp256r1 (eq. 3072 bits RSA) FS</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0x8030) ECDH secp256r1 (eq. 3072 bits RSA) FS</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0x802f) ECDH secp256r1 (eq. 3072 bits RSA) FS</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc024) ECDH secp256r1 (eq. 3072 bits RSA) FS</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc023) ECDH secp256r1 (eq. 3072 bits RSA) FS</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc014) ECDH secp256r1 (eq. 3072 bits RSA) FS</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027) ECDH secp256r1 (eq. 3072 bits RSA) FS</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_256_CBC_SHA (0x3d)</td>
<td>256</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)</td>
<td>128</td>
<td></td>
</tr>
</tbody>
</table>

**Highlighted Cipher Suites:**

- TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xccc5) ECDH secp256r1 (eq. 3072 bits RSA) FS
- TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xccc8) ECDH secp256r1 (eq. 3072 bits RSA) FS

**Notes:**

- **FS:** Full Strength
- **256:** 256-bit key size
- **128:** 128-bit key size
- **??:** Indicates a question mark
- **256P:** 256-bit key size
- **WEAK:** Ciphers marked as weak

**8**
Non-FIPS Ciphers – What Happened?

• Developers installed a FIPS-validated crypto library
  - “FIPS Inside” – product can legitimately claim to have a FIPS certificate number

• Never enabled FIPS mode…
  - OpenSSL: “FIPS_mode_set()”

• Never edited application config files to disable non-FIPS ciphers
  - Applications would crash if FIPS mode enabled

• Supplementary certification testing like CC or UC-APL DODIN-APL would catch this
Example 5 – Linux OS Complexity

• We have an “appliance” that runs on CentOS
• We replace OpenSSL with a FIPS-validated crypto library (e.g. SafeLogic)
• Q: Is everything cool?

• A: Complicated…
Example 6 – More Linux OS Complexity

- Is this FIPS-validated crypto?
- Does it need to be?
9.2.1. Enabling FIPS Mode

To make Red Hat Enterprise Linux 6 compliant with the Federal Information Processing Standard (FIPS) Publication 140-2, you need to make several changes to ensure that certified cryptographic modules are used. To turn your system (kernel and user space) into FIPS mode, follow these steps:

1. For proper operation of the in-module integrity verification, the prelink has to be disabled. This can be done by setting `PRELINKING=no` in the `/etc/sysconfig/prelink` configuration file. Existing prelinking, if any, should be undone on all system files using the `prelink -d -a` command.

2. Next, install the dracut-fips package:

   ```bash
   # yum install dracut-fips
   ```

3. Recreate the `initsrd` file (this operation will overwrite the existing `initsrd` file):

   ```bash
   # dracut -f
   ```

4. Modify the kernel command line of the current kernel in the `/boot/grub/grub.conf` file by adding the following option:

   ```
   fips=1
   ```

   If the `/boot` or `/boot/efi` directories are located on a separate partition, the `boot=partition` kernel parameter must be added to the kernel command line. Replace `partition` with the partition that contains the `/boot` or `/boot/efi` directory. Partitions can be identified using the `df` command. For example:

   ```
   # df /boot
   ```

- **Must enable FIPS mode in kernel**

- **Packages such as OpenSSH read this value**

- **So does kernel crypto (e.g. disk encryption)**

- **Q: Is this FIPS validated?**

  - **A: No. Not in CentOS.**

    - … but it’s “FIPS compliant” so you might fool everyone.
A Sane Approach?

• This is really difficult to get perfect – especially on modern Linux-based web applications

• Identify services that need to be FIPS validated. Make sure they are.
  - E.g. for a network product – SSH, IPsec, Wi-Fi, SNMP
  - E.g. for a file server – disk encryption
  - E.g. for a database – database encryption

• Be clear in your marketing what is and isn’t covered by your FIPS certificate
  - … or don’t. Your FIPS security policy will give all these details. Anything not there can be presumed not to be covered. Buyer beware.
THANK YOU

a Hewlett Packard Enterprise company