aruba
a Hewlett Packard Enterprise company

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…**

```
# 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?**

```
17  /*TODO call this routine to update the GMK*/
18  void
19  new_gmk()
20  {
21
22  }
```

```
17  /* call this routine to update the GMK*/
18  void
19  new_gmk(struct sap_info *sap)
20  {
21      generate_random(sap->gmk, sizeof (sap->gmk));
22      return;
23  }
```

Example 3 – Large Multi-Module Products

- **Is this concerning?**

```
3  import java.io.ByteArrayInputStream;
4  import java.security.cert.CertificateFactory;
5  import java.security.cert.X509Certificate;
6  import java.util.ArrayList;
7  import java.util.List;
```

```
22 public class CATrustedStore {
23
24         private static final Log log = LogFactory.getLog(CATrustedStore.class);
25         private static CATrustedStore instance = new CATrustedStore();
26         private List<X509Certificate> serverCTL = new ArrayList<>();
27         private List<X509Certificate> caCTL = new ArrayList<>();
```

- **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

Cipher Suites

 # TLS 1.2 (suites in server-preferred order)

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c) ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b) ECDH secp256r1 (eq. 3072 bits RSA) FS	128
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030) ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f) ECDH secp256r1 (eq. 3072 bits RSA) FS	128
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024) ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023) ECDH secp256r1 (eq. 3072 bits RSA) FS	128
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028) ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014) ECDH secp256r1 (eq. 3072 bits RSA) FS	256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027) ECDH secp256r1 (eq. 3072 bits RSA) FS	128
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013) ECDH secp256r1 (eq. 3072 bits RSA) FS	128
TLS_RSA_WITH_AES_256_GCM_SHA384 (0x9d)	256
TLS_RSA_WITH_AES_128_GCM_SHA256 (0x9c)	128
TLS_RSA_WITH_AES_256_CBC_SHA256 (0x3d)	256
TLS_RSA_WITH_AES_128_CBC_SHA256 (0x3c)	128
TLS_RSA_WITH_AES_128_CBC_SHA (0x2f)	128
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9) ECDH secp256r1 (eq. 3072 bits RSA) FS	256^{P}
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8) ECDH secp256r1 (eq. 3072 bits RSA) FS	256^{P}
TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xa) **WEAK**	112

??

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

```
root:$6$QcJxcu$Qgz9LmICgfw/UdxGHpZpzVylZQZ/cXe6WRC5BArZVFfGpL/Hsrla6w06GSUjhePbOqpzF5JOf14/A6MAJVr9A/::0:99999:7:::
bin:*:17110:0:99999:7:::
daemon:*:17110:0:99999:7:::
adm:*:17110:0:99999:7:::
lp:*:17110:0:99999:7:::
sync:*:17110:0:99999:7:::
shutdown:*:17110:0:99999:7:::
halt:*:17110:0:99999:7:::
mail:*:17110:0:99999:7:::
operator:*:17110:0:99999:7:::
games:*:17110:0:99999:7:::
ftp:*:17110:0:99999:7:::
nobody:*:17110:0:99999:7:::
systemd-bus-proxy:!!:17279::::::
systemd-network:!!:17279::::::
dbus:!!:17279::::::
polkitd:!!:17279::::::
tss:!!:17279::::::
sshd:!!:17279::::::
postfix:!!:17279::::::
chrony:!!:17279::::::
jon:$6$m8D4J$mYAON8uq5.McTfn8KE2KXKNfiTMkUf4.eZCTkNGaW6dZO5z9oBiJeNatLt5uJ80F2fVi4ZOZTN7bNUEUM1:17279:0:99999:7:::
rpc:!!:17279:0:99999:7:::
saslauth:!!:17279::::::
radvd:!!:17279::::::
qemu:!!:17279::::::
rpcuser:!!:17279::::::
nfsnobody:!!:17279::::::
unbound:!!:17279::::::
```

- **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 -u -a` command.

2. Next, install the dracut-fips package:

```
~]# yum install dracut-fips
```

Note

FIPS integrity verification is performed when the dracut-fips package is present on the system, regardless of whether the system operates in FIPS mode or not. However, the integrity verification results are ignored (or only logged) if the system or a shared library is not in FIPS mode, even when dracut-fips is present.

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

```
~]# 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**

```
CODE: SELECT ALL

$ cat /proc/sys/crypto/fips_enabled
1
```

- **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