**Début analyse**

Initialisation

* Predictable sequences (12)
  + CVE-2011-0766 : <https://www.vulncode-db.com/CVE-2011-0766>
    - Erlang && C
      * Problem:
        + **ssh\_bits.erl**

**irandom(N, Top, Bottom)**

**random(N):**

* + - * + **ssh\_connection\_handler.erl**

**{A,B,C} = erlang:now()**

**random:seed(A, B, C)**

* + - * Fix:
        + **ssh\_connection\_handler.erl**

**Remove {A,B,C} = erlang:now()**

**Remove random:seed(A, B, C)**

* + - * + **crypto.c**

**strong\_rand\_bytes (ajouté)**

**strong\_rand\_mpint (ajouté)**

* + - * + **ssh\_bits.erl**

random(N) : **strong\_rand\_bytes**

irandom(N, Top, Bottom) : **strong\_rand\_mpint**

strong\_rand\_test (test low entropy, bonus )

* + CVE-2012-2417 : <https://www.vulncode-db.com/CVE-2012-2417>
    - Python
      * Problem:
        + **ElGamal.py**

**obj.g=bignum(getPrime(size, randfunc))**

* + - * Fix:
        + **ElGamal.py**

**if safe and divmod(obj.p-1, obj.g)[1]==0**

**if safe and divmod(obj.p-1, ginv)[1]==0:**

* + CVE-2013-1445 : <https://www.vulncode-db.com/CVE-2013-1445>
    - Python
      * Problem:
        + **from Crypto.Random import OSRNG**
        + **from Crypto.Random.Fortuna import FortunaAccumulator**
        + Using Crypto.Random.atfork()

Read from the Crypto.Random PRNG, causing an **internal reseed**.

Fork the process and invoke Crypto.Random.atfork() in the child.

Read from the Crypto.Random PRNG again, in at least two different processes (parent and child, or multiple children).

* + - * Fix:
        + **FortunaAccumulator.py**

**def \_forget\_last\_reseed(self):**

* + - * + **\_UserFriendlyRNG.py**

**self.\_fa.\_forget\_last\_reseed()**

* + CVE-2014-5386 : <https://www.vulncode-db.com/CVE-2014-5386>
    - C++
      * Problem:
        + **ext\_mcrypt.cpp**

iv[--size] = (char)(255.0 \* **rand()** / RAND\_MAX);

* + - * Fix:
        + **ext\_mcrypt.cpp**

**#include "hphp/runtime/ext/ext\_math.h"**

iv[--size] = (char)**f\_rand**(0, 255);

* + CVE-2015-8867 : <https://www.cvedetails.com/cve/CVE-2015-8867/>
    - C
      * Problem:
        + It was discovered that the PHP openssl\_random\_pseudo\_bytes() function did not return cryptographically strong pseudo-random bytes
        + **openssl.c**

if ((strong\_result = RAND\_pseudo\_bytes(buffer, buffer\_length)) < 0) {

* + - * Fix:
        + **openssl.c**

if (**RAND\_bytes**(buffer, buffer\_length) <= 0) {

* + CVE-2018-12520 : <https://www.cvedetails.com/cve/CVE-2018-12520/>
    - C++
      * Problem:
        + PRNG involved in the generation of session IDs is not seeded at program startup
        + **HTTPserver.cpp**
      * Fix:
        + **HTTPserver.cpp**

struct timeval tv;

/\* Randomize data \*/

gettimeofday(&tv, NULL);

**srand**(tv.tv\_sec + tv.tv\_usec)

* + CVE-2019-11808 : <https://www.cvedetails.com/cve/CVE-2019-11808/>
    - Java
      * Problem:
        + java.util.concurrent.ThreadLocalRandom (not cryptographically secure). Could have been avoided if programmer read documentation.
        + **DefaultSessionIdGenerator.java**

public AsciiString generateSessionId()

**java.util.concurrent.ThreadLocalRandom**

* + - * Fix:
        + **DefaultSessionIdGenerator.java**

public AsciiString generateSessionId()

**java.util.UUID**

* + CVE-2020-12735 : <https://github.com/domainmod/domainmod/issues/122>
    - PHP
      * Problem:
        + **reset.php**

$new\_password = substr(md5(**time()**), 0, 8);

* + - * Fix:
        + **reset.php**

$new\_password = $user->**generatePassword(30)**;

$new\_hash = $user->generateHash($new\_password);

* + CVE-2020-28924 : <https://github.com/rclone/rclone/issues/4783>
    - GO
      * Problem: Utilisation des fonctions dans **: math/rand** à la place de **crypto/rand**

**random.go**

out[i] = source[rand.Intn(len(source))]

**math/rand**

n, err := rand.Read(pw)

**math/rand**

**math/rand (librairie)**

func read(p []byte, src Source, readVal \*int64, readPos \*int8) (n int, err error)

**crypto/rand (librairie)**

func Read(b []byte) (n int, err error)

* + - * Fix:
        + func Password(bits int) (password string, err error) uses rand from **crypto/rand**

**random.go**

out[i] = source[mathrand.Intn(len(source))]

**math/rand**

n, err := cryptorand.Read(pw)

**crypto/rand**

* + - * + To help add entropy (but not fix):

**random.go**

func Seed() error

**math/rand**

* + CVE-2021-3538 : <https://www.cvedetails.com/cve/CVE-2021-3538/>
    - GO
      * Problem:
        + Lorsque Read rencontre une erreur ou une condition de fin de fichier après avoir lu avec succès n > 0 octets, il renvoie le nombre d'octets lus.
        + **generator.go**

if \_, err := g.rand.Read(u[:]); err != nil {

* + - * Fix:
        + ReadFull lit exactement len(buf) octets de r (reader) dans buf. Il renvoie le nombre d'octets copiés et une erreur si moins d'octets ont été lus.
        + **generator.go**

if \_, err := io.ReadFull(g.rand, u[:]); err != nil {

* + CVE-2021-41117 : <http://m.cvedetails.com/cve/CVE-2021-41117/>
    - Javascript
      * Problem:
        + **index.js**

b.putByte(String.fromCharCode(next & 0xFF))

* + - * Fix:
        + **index.js**

b.putByte(next & 0xFF);

* + CVE-2022-36045 : <https://www.cvedetails.com/cve/CVE-2022-36045/>
    - Javascript
      * Problem:
        + Math.random() (not cryptographically secure). Could have been avoided if programmer read documentation.
        + **src\_utils.js (both)**
        + **public\_src\_utils.js (v.1.19.x)**

**Math.random**

* + - * + **public\_src\_utils.common.js (v.2.x)**

**Math.random**

* + - * Fix:
        + **src\_utils.js (both)**

require('crypto');

* + - * + **public\_src\_utils.js (v.1.19.x)**
        + **public\_src\_utils.common.js (v.2.x)**
* Re-use (2)
  + CVE-2019-15075 : <https://www.cvedetails.com/cve/CVE-2019-15075/>
    - PHP
      * Problem:
        + **config.php**

$config ['private\_key'] = '8YSDaBtDHAB3EQkxPAyTz2I5DttzA9uR';

$config ['encryption\_key'] = 'r)fddEw232f';

* + - * Fix:
        + **config.php**

$config ['private\_key'] = $astpp\_config ['PRIVATE\_KEY'];

$config ['encryption\_key'] = $astpp\_config ['ENCRYPTION\_KEY'];

* + CVE-2022-1434 : <https://www.cvedetails.com/cve/CVE-2022-1434/>
    - C
      * Problem:
        + Une erreur de copier-coller signifiait que le chiffrement RC4-MD5 (utilisé dans TLS) utilisait les données AAD (Additionnal Authentication Data?) de TLS comme clé MAC.
        + **cipher\_rc4\_hmac\_md5.c**

p = OSSL\_PARAM\_locate\_const(params, **OSSL\_CIPHER\_PARAM\_AEAD\_TLS1\_AAD**);

* + - * Fix:
        + **cipher\_rc4\_hmac\_md5.c**

p = OSSL\_PARAM\_locate\_const(params, **OSSL\_CIPHER\_PARAM\_AEAD\_MAC\_KEY**);

* Weak values (2)
  + CVE-2019-10908 : <https://github.com/airsonic/airsonic/commit/61c842923a6d60d4aedd126445a8437b53b752c8>
    - Java
      * Problem:
        + This PRNG has a 48-bit seed that can easily be bruteforced, leading to trivial privilege escalation attacks (org.apache.commons.lang.RandomStringUtils)
        + **RecoverController.java**

**Impot org.apache.commons.lang.RandomStringUtils**

**import java.util.Random; (not cryptographically secure, 48-bit seed)**

String password = RandomStringUtils.randomAlphanumeric(8);

* + - * Fix:
        + **RecoverController.java**

**import java.security.SecureRandom; (up to 128-bit seed)**

int index = random.nextInt(SYMBOLS.length());

* + CVE-2022-1235 : <https://github.com/livehelperchat/livehelperchat/commit/6538d6df3d8a60fee254170b08dd76a161f7bfdc>
    - PHP
      * Problem:
        + **lhc\_web\cli\lib\install.php**

$cfgSite->setSetting( 'site', 'secrethash', **substr(md5(time() . ":" . mt\_rand()),0,10)**);

* + - * + **lhc\_web\modules\lhinstall\install.php**

$cfgSite->setSetting( 'site', 'secrethash', (!empty(getenv('LHC\_SECRET\_HASH')) ? getenv('LHC\_SECRET\_HASH') : **substr(md5(time() . ":" . mt\_rand()),0,10)**));

* + - * Fix:
        + **lhc\_web\cli\lib\install.php**

$cfgSite->setSetting( 'site', 'secrethash', **erLhcoreClassChat::generateHash(80)**)

* + - * + **lhc\_web\modules\lhinstall\install.php**

$cfgSite->setSetting( 'site', 'secrethash', (!empty(getenv('LHC\_SECRET\_HASH')) ? getenv('LHC\_SECRET\_HASH') : **erLhcoreClassChat::generateHash(80)**));

Insecure defaults (1)

* + CVE-2016-1000352&1000344 :
    - <https://www.cvedetails.com/cve/CVE-2016-1000352/>
    - <https://www.cvedetails.com/cve/CVE-2016-1000344/>
    - Java
      * Problem:
        + **dh\_IESCipher.java && ec\_IESCipher.java**

import org.bouncycastle.crypto.engines.**AESEngine**;

* + - * + **AESEngine.java**

**returns an AESEngine that uses AES ECB cipher mode**

* + - * Fix:
        + **dh\_IESCipher.java && ec\_IESCipher.java**

import org.bouncycastle.crypto.engines.**AESFastEngine**;

* + - * + **AESFastEngine.java**

**Does not default to ECB mode.**

Weak algorithms (1)

* + CVE-2012-3458 : [https://www.vulncode-db.com/CVE-2012-3458](%20https://www.vulncode-db.com/CVE-2012-3458)
    - Python
      * Problem:
        + PyCrypto to encrypt sessions, uses **AES in ECB cipher mode (default)**
        + **pycrypto.py**

cipher = **AES.new(key)**

data = data + (" " \* (16 - (len(data) % 16)))

* + - * Fix:
        + **pycrypto.py**

**cipher = AES.new(key, AES.MODE\_CTR,**

**counter=Counter.new(128, initial\_value=0))**

Validation (3)

* + CVE-2016-2053 : <https://www.vulncode-db.com/CVE-2016-2053>
    - C
      * Problem:
        + Une clé avec des traits spécifique pouvait être créé pour déclencher BUG\_ON() et provoquer une panique du noyau et planter le système.
        + **asn1\_decoder.c**

if ((op & ASN1\_OP\_MATCH\_\_COND && flags & FLAG\_MATCHED) || dp == datalen) {

* + - * Fix:
        + **asn1\_decoder.c**

if ((op & ASN1\_OP\_MATCH\_\_COND && flags & FLAG\_MATCHED) || (**op & ASN1\_OP\_MATCH\_\_SKIP** && dp == datalen)) {

* + CVE-2019-11578 : <https://www.cvedetails.com/cve/CVE-2019-11578/>
    - C
      * Problem:
        + auth.c in dhcpcd before 7.2.1 allowed attackers to infer secrets by performing latency attacks.
        + **auth.c**

if (**memcmp**(d, &hmac\_code, dlen)) {

* + - * Fix:
        + **auth.c**

if (!**consttime\_memequal**(d, &hmac\_code, dlen)) {

* + CVE-2021-32738 : <https://github.com/stellar/js-stellar-sdk/compare/v8.2.2...v8.2.3>
    - Typescript (Javascript)
      * Problem:
        + La fonction readChallengeTx ne **vérifie pas** que **le serveur** a **signé** la transaction
        + **utils.ts**

readChallengeTx

Aucune vérification de signature du serveur.

* + - * Fix:
        + **utils.ts**

readChallengeTx

**if (!verifyTxSignedBy(transaction, serverAccountID)) {**

**throw new InvalidSep10ChallengeError(**

**`Transaction not signed by server: '${serverAccountID}'`,**

**);**

**}**

Usage Complexity (5)

* + CVE-2017-7526 : <https://www.cvedetails.com/cve/CVE-2017-7526/>
    - C
      * Problem:
        + Vulnerable to a cache side-channel attack resulting into a complete break of RSA-1024 while using the left-to-right method for computing the sliding-window expansion
        + **rsa.c**

**secret\_core\_crt()**

* + - * Fix:
        + **rsa.c**

**secret\_core\_crt() : Exponant blinding (de la clé privé *d*)**

* + CVE-2018-16870 : <https://www.cvedetails.com/cve/CVE-2018-16870/>
    - C
      * Problem:
        + Vulnerable to a new variant of the Bleichenbacher attack to perform downgrade attacks against TLS.
        + **rsa.c**

**static int RsaUnPad(const byte \*pkcsBlock, unsigned int pkcsBlockLen, byte \*\*output, byte padValue)**

* + - * Fix:
        + **rsa.c**

**static int RsaUnPad(const byte \*pkcsBlock, unsigned int pkcsBlockLen, byte \*\*output, byte padValue)**

**Minimum of 11 bytes of pre-message data and must have separator**

* + CVE-2018-19653 : <https://www.cvedetails.com/cve/CVE-2018-19653/>
    - Go
      * Problem:
        + **verify\_server\_hostname - If set to true**, **Consul verifies for all outgoing TLS connections that the TLS certificate presented by the servers matches "server**. From versions 0.5.1 to 1.4.0, due to a bug, setting this flag alone **does not** imply verify\_outgoing and leaves client to server and server to server RPCs ***unencrypted*** despite the documentation stating otherwise.
        + **config.go**

// If VerifyServerHostname is true, that implies **(vulnérabilité ici, mauvaise documentation) VerifyOutgoing**

* + - * Fix:
        + **config.go**

verifyServerName := b.boolVal(c.VerifyServerHostname)

verifyOutgoing := b.boolVal(c.VerifyOutgoing)

if verifyServerName {

**verifyOutgoing = true**

}

* + CVE-2019-9155 : <https://www.cvedetails.com/cve/CVE-2019-9155/>
    - Javascript
      * Problem:
        + The implementation of the Elliptic Curve Diffie-Hellman (ECDH) key exchange algorithm does not **verify** that the communication partner's public key is **valid** (i.e., that the point lies on the elliptic curve). This causes the application to implicitly calculate the resulting secret key not based on the **specified elliptic curve** but rather **an altered curve**.
        + **ecdh.js**

**async function kdf(hash\_algo, X, length, param)**

**Manque la curve (un autre curve est dérivé)**

**Affecte les fonctions decrypt et encrypt**

* + - * Fix:
        + **ecdh.js**

async function kdf(hash\_algo, S, length, param, **curve**, **compat**)

* + CVE-2020-26263 : <https://www.cvedetails.com/cve/CVE-2020-26263/>
    - Python
      * Problem:
        + The code that performs decryption and padding check in RSA PKCS#1 v1.5 decryption is data dependant (**multiple ways in which it leaks information**)
        + **0.7.6\_rsakey.py && 0.8.0-alpha39\_rsakey.py**

**def decrypt(self, encBytes)**

* + - * Fix:
        + **0.7.6\_rsakey.py && 0.8.0-alpha39\_rsakey.py**

**def decrypt(self, encBytes)**

Other (4)

* + CVE-2013-2548 : <https://www.vulncode-db.com/CVE-2013-2548>
    - C
      * Problem:
        + Voir document Word pour une bonne description.
      * Fix:
        + switch **snprintf() 🡪 strncpy()**
        + switch **memcpy() 🡪 strncpy()**
        + **Use length of module name instead of CRYPTO\_MAX\_ALG\_NAME**
        + **Initialize** ualg cru\_type && ualg\_cru\_mask
  + CVE-2014-3570 : <https://www.vulncode-db.com/CVE-2014-3570>
    - C
      * Problem: Voir documentation dans le répertoire respectif (complexe)
        + **bn\_asm.c**

BN\_LLONG, **BN\_UMULT\_LOHI**, **BN\_UMULT\_HIGH**, **!BN\_LLONG**

mul\_add\_c(a,b,c0,c1,c2)

**mul\_add\_c2(a,b,c0,c1,c2)**

sqr\_add\_c(a,i,c0,c1,c2)

* + - * + **bntest.c**
        + **mips.pl**
        + **x86\_64-gcc.c**
      * Fix:
        + **bn\_asm.c**

BN\_LLONG, **BN\_UMULT\_LOHI**, **BN\_UMULT\_HIGH**, **!BN\_LLONG**

mul\_add\_c(a,b,c0,c1,c2)

**mul\_add\_c2(a,b,c0,c1,c2)**

sqr\_add\_c(a,i,c0,c1,c2)

* + - * + **bntest.c**
        + **mips.pl**
        + **x86\_64-gcc.c**
  + CVE-2014-8275 : <https://www.vulncode-db.com/CVE-2014-8275>
    - C
      * Problem:
        + Does not enforce certain constraints on certificate data which allows **attackers to include crafted data** within a certificate's unsigned portion.
        + **a\_verify.c**

**Mauvaise variable 🡪 mauvais code de fonction utilisé**

**ASN1\_F\_ASN1 \_VERIFY** 🡪 **ASN1\_F\_ASN1\_ITEM\_VERIFY**

* + - * + **dsa\_asn1.c && ecs\_vrf.c**

**Aucune vérification interne de la portion non signée du certificat.**

* + - * + **x\_all.c**

**Ne vérifie pas si l’encodage** **de l’algorithme de signature est identique à celui du certificat.**

* + - * Fix:
        + **a\_verify.c**

**ASN1err(ASN1\_F\_ASN1\_ITEM\_VERIFY, ASN1\_R\_INVALID\_BIT\_STRING\_BITS\_LEFT);**

* + - * + **dsa\_asn1.c**

**if (derlen != siglen || memcmp(sigbuf, der, derlen))**

* + - * + **ecs\_vrf.c**

**if (derlen != sig\_len || memcmp(sigbuf, der, derlen))**

* + - * + **x\_all.c**

**if (X509\_ALGOR\_cmp(a->sig\_alg, a->cert\_info->signature)) return 0;**

* + CVE-2016-10530 : <https://www.cvedetails.com/cve/CVE-2016-10530/>
    - Javascript
      * Problem:
        + Defaults to sending environment variables over **HTTP** (instead of **HTTPS**)
        + **airbrake.js**

this.host = '**http**://' + os.hostname();

this.protocol = '**http**';

* + - * Fix:
        + **airbrake.js**

this.host = '**https**://' + os.hostname();

this.protocol = '**https**';

* C = 10
* Javascript = 5 (CVE-2021-32738 is technically typescript)
* Python = 4
* PHP = 3
* GO = 3
* Java = 3
* C++ = 2
* Erlang = 1 (can be included with C since CVE-2011-0766 uses both languages)