**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 (2)

* + 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))**

* + 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.**

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)
  + ***Predictable sequence (12) C = 3 Erlang =1 Python = 2 C++ = 2 Java = 2 PHP = 3 Go = 2 JS = 2***

Initialisation

* Predictable sequences (12)
  + CVE-2011-0766 : Erlang && C
    - Problem: Le PRNG générer est basé sur le temps (fonction *now()*).
    - Fix : Utiliser un autre moyen que le temps pour générer le PRNG.
  + CVE-2012-2417 : Python
    - Problem: Algorithme de chiffrement à clé asymétrique. Utilise un sous ensemble à la place de l’ensemble qui est supposé utilisée
    - Fix : Vérifier que le générateur ne divise pas p-1. Si il le divise, recalculer
  + CVE-2013-1445 : Python
    - Problem: Lorsqu’un thread est généré par la fonction Crypto.Random.atfork(), le seed utilisée est identique au thread parent. Ceci rend le PRNG prédictible.
    - Fix: Effacer le dernier seed utiliser lorsqu’un nouveau thread qui utilise la fonction Crypto.Random.atfork() est générée.
  + CVE-2014-5386 : C++
    - Problem: Aucun seed est utilisée lors de la génération de nombres aléatoires, en conséquence le vecteur d’initialisation est toujours identique.
    - Fix: Utilise rune librairie qui seed automatiquement lors de la génération de nombres aléatoires.
  + CVE-2015-8867 : C
    - Problem: Utilisation d'une méthode qui n'est pas cryptographiquement sécurisée à des fins cryptographiques. De plus, cette fonction est obselète. **RAND\_pseudo\_bytes**
    - Fix: Utilisé une méthode qui est cryptographiquement sécurisé et non obsolète. **RAND\_bytes**
  + CVE-2018-12520 : C++
    - Problem: PRNG impliqué dans la génération des identifiants de session n'est pas amorcé au démarrage du programme
    - Fix: Amorcé le générateur de nombres aléatoires utilisé par la fonction rand : function void **srand**(unsigned int seed)
  + CVE-2019-11808 : Java
    - Problem: Utilisation d’une function qui n’est pas cryptographiquement secure : ThreadLocalRandom
    - Fix: Utilisée une fonctione qui est cryptographiquement secure : randomUUID()
  + CVE-2020-12735 : PHP
    - Problem: Utilise le temps pour dérivé un mot de passe lorsque le mot de passe est réinitialisé.
    - Fix: Utiliser une fonction qui génère suffisamment d'entropie (generatePassword())
  + CVE-2020-28924 : GO
    - Problem: Utilisation des fonctions dans **: math/rand** à la place de **crypto/rand**
    - Fix: Utilisée **crypto/rand** à des fins cryptographique
  + CVE-2021-3538 : GO
    - Problem : Utilisation d’une fonction qui gère les certaines situations d’une façon problématique : rand.Read().
    - Fix: Utiliser une fonctione qui gère ces situations de la façon secucre : ReadFull() lit exactement len(buf) octets de r (reader) dans le tampon.
  + CVE-2021-41117 : Javascript
    - Problem: Mauvaise utilisation de paramètre dans l’appelle d’une fonction (le paramètre entrée est une étape qui se fait déjà à l’intérieur de la fonction putByte).
    - Fix: Enlever la section qui est problématique (pour ne pas répéter cette opération deux fois) : String.fromCharCode
  + CVE-2022-36045 : Javascript
    - Problem: L’utilisation d’une function qui n’est pas cryptographiquement secure (Math.random()). Cette erreur aurait pû être évité car dans la documentation de Math.random(), c’est mentionné que la fonction n’est pas cryptographiquement secure.
    - Fix: Utilisation d’une librairie qui offre des fonctions cryptographiquement secure : Node.js Crypto Module
* Re-use (2)
  + CVE-2019-15075 : PHP
    - Problem: Réutilisation des même clés qui ne sont pas aléatoirement fortes.
    - Fix: Utiliser des clés aléatoires fortes.
  + CVE-2022-1434 : C
    - Problem: Une erreur de copier-coller qui a fait en sorte que le chiffrement RC4-MD5 (utilisé dans TLS) utilisait les données AAD (Additionnal Authentication Data) de TLS comme clé MAC.
    - Fix: Remplacer les données erronées par les bonne données.
* Weak values (2)
  + CVE-2019-10908 : Java
    - Problem: Utilisation d’une fonction qui utilise un seed de 48 bits qui peut facilement être brisée par une recherche exhaustive. **Import org.apache.commons.lang.RandomStringUtils**
      * **import java.util.Random**
    - Fix: Utiliser une fonctione qui génère des nombre aléatoire utilisant un seed avec de la sécurité suffisante (import java.security.SecureRandom (128-bit seed)
  + CVE-2022-1235 : PHP
    - Problem: Utilise une fonctione qui génère une valeur qui peut prendre maximum  valeurs (facile à briser par recherche exhaustive).
    - Fix: Augmenter le nombre de valeurs possible (dans ce cas, ).

Insecure defaults (2)

* + CVE-2012-3458 : Python
    - Problem: Lorsque la creation d’une nouvelle clé est fait sans spécifier un deuxième paramètre, la valeur par défaut est le mode de chiffrement AES ECB (pas cryptographiquement secure) : **AES.new(key)**
    - Fix: Ajouter les paramètres nécessaire pour assurer de ne pas tomber sur la valeur par défaut : AES.new(key, AES.MODE\_CTR, counter=Counter.new(128, initial\_value=0))
  + CVE-2016-1000352&1000344 : Java
    - Problem: Dans La classe AESEngine utilise par défaut le mode de chiffrement AES ECB (pas cryptographiquement secure) : import org.bouncycastle.crypto.engines.AESEngine
    - Fix: Changer la classe qui n’est pas en mode ECB par défaut (import org.bouncycastle.crypto.engines.**AESFastEngine**)

Validation (3)

* + CVE-2016-2053 : C
    - Problem : Il manquait une condition à vérifier pour ne pas déclencher la sécurité intégrer BUG\_ON(), ceci plantait le système. Le bug se produit lors de la création de la clé (manque une étape de validation dans le système conçu). C’est un problème spécifique au produit Linux.
    - Fix: Ajouter la condition pour ne pas déclencher cette mesure de sécurité : **op & ASN1\_OP\_MATCH\_\_SKIP** (équivaut à 0x01).
  + CVE-2019-11578 : C
    - Problem: Utilise une function qui est vulnérable aux attaques de latence : memcmp(const void \*str1, const void \*str2, size\_t n)
    - Fix: Utilise rune fonction qui assure un traitement de données en temps constant : consttime\_memequal(const void \*b1, const void \*b2, size\_t len) (pour ne pas donner de fuites d’informations).
  + CVE-2021-32738 : Typescript (Javascript)
    - Problem: La fonction readChallengeTx ne **vérifie pas** que **le serveur** a **signé** la transaction.
    - Fix: Ajouter une condition qui vérifie si la transactions est signée : **if (!verifyTxSignedBy(transaction, serverAccountID))**

Usage Complexity (5)

* + CVE-2017-7526 : C
    - Problem: Vulnérable à une attaque par canal latéral du cache entraînant une rupture complète de RSA-1024 lors de l'utilisation de la méthode de gauche à droite pour calculer l'expansion de la fenêtre glissante.
    - Fix: Obfusqué la clé privée en utilisant une technique qu’on nomme **: Exponant blinding**.
  + CVE-2018-16870 : C
    - Problem: Vulnérable à une variante de l'attaque Bleichenbacher pour effectuer des attaques de déclassement contre TLS. C’est dû a un manque de vérification d’un nombre de bytes minimum lors de la décompression de la clé RSA.
    - Fix: Ajouter une verification qui vérifie qu’un paquet à un minimum de 11 octets de données de pré-message incluant le séparateur : if (**i < RSA\_MIN\_PAD**\_SZ || pkcsBlock[i-1] != 0) {
  + CVE-2018-19653 : Go
    - Problem: Une mauvaise documentation est liée au problème. Dans la documentation, c’est mentionner qu’activer un paramètre (**verify\_server\_hostname**) implique un autre paramètre (qui est nécessaire pour assurer l’encryption de la communication entre serveurs). Ceci n’est pas le cas.
    - Fix: Dans la fichier parent (à importer), le paramètre qui devait être initialisée à true par l’utilisation de l’autre variable a été explicitement initialiser : **verifyOutgoing = true**.
  + CVE-2019-9155 : Javascript
    - Problem: La mise en œuvre de l'algorithme d'échange de clés Elliptic Curve Diffie-Hellman (ECDH) ne vérifie pas que la clé publique du partenaire de communication est valide (c'est-à-dire que le point se trouve sur la courbe elliptique). Cela amène l'application à calculer implicitement la clé secrète résultante non basée sur la courbe elliptique spécifiée mais plutôt sur une courbe modifiée : **kdf(hash\_algo, X, length, param)**
    - Fix: Ajout du paramètre dans l’appel de fonction qui fait la dérivation des clés pour qu’elle utilise la courbe elliptique spécifiée : kdf(hash\_algo, S, length, param, **curve**, **compat**)
  + CVE-2020-26263 : Python
    - Problem: Le code qui effectue le déchiffrement et la vérification du remplissage dans le déchiffrement RSA PKCS#1 v1.5 dépend des données. Le traitement de données cause des fuites d’informations  : if decBytes[0] != 0 or decBytes[1] != 2: return None
    - Fix: Renvoie un message aléatoire sélectionné de manière déterministe au cas où la vérification du remplissage de bits échoue.

Other (4)

* + CVE-2013-2548 : C
    - Problem: Utilisation d’une fonction qui ne gère pas bien le transfert de bits : **snprintf()**
    - Fix: Utiliser une fonction qui gère bien le transfert de bits : **strncpy()**
  + CVE-2014-3570 : C
    - Problem: Dans le calcul de BigNumbers, une erreur se produit avec une probabilité de 1/ sur une plateforme d’architecure MIPS de 32 bits ou une probabilité de 1/ sur une plateforme de 64 bits. C’est vrai que les probabilités sont extrêmement improbables, néanmoins, nous avons couvert cette vulnérabilité de toute façon. Le problème est causé lorsque l’addition du gros-boutiste et du petit-boutiste est plus grand que . Le bit carry n’est pas pris en considération et ceci cause un débordement de tampon. Cette erreur pourrais en théorie donner la chances aux attaquants distants non authentifiés de déjouer les mécanismes de protection cryptographique via des vecteurs non spécifiés.
    - Fix: Faire en sorte que l’addition du gros-boutiste et du petit-boutiste ne peut jamais être plus grand que . Ceci ce fait en ayant une variable qui gère la situation qui est extrêmement improbable…
  + CVE-2014-8275 : C
    - Problem: N'applique pas certaines contraintes sur les données de certificat, ce qui permet aux attaquants d'inclure des données spécialement conçues dans la partie non signée d'un certificat (Aucune vérification interne de la portion non signée du certificat. Aucune vérification sur l’encodage de l’algorithme de signature pour valider si elle est identique à celui du certificat).
    - Fix: Ajout des variables et conditions qui font les vérifications nécessaire pour ne pas permettre à un attaquants d’inclure des données dans la partie non signée : **if (derlen != siglen || memcmp(sigbuf, der, derlen)) . if (X509\_ALGOR\_cmp(a->sig\_alg, a->cert\_info->signature)) return 0;**
  + CVE-2016-10530 : Javascript
    - Problem: Utilisation de HTTP (aucune encryption faites sur les données en transmission.
    - Fix: Utiliser HTTPS (encrypte les données lors de la transmission).