Challenge Brigitte Friang.

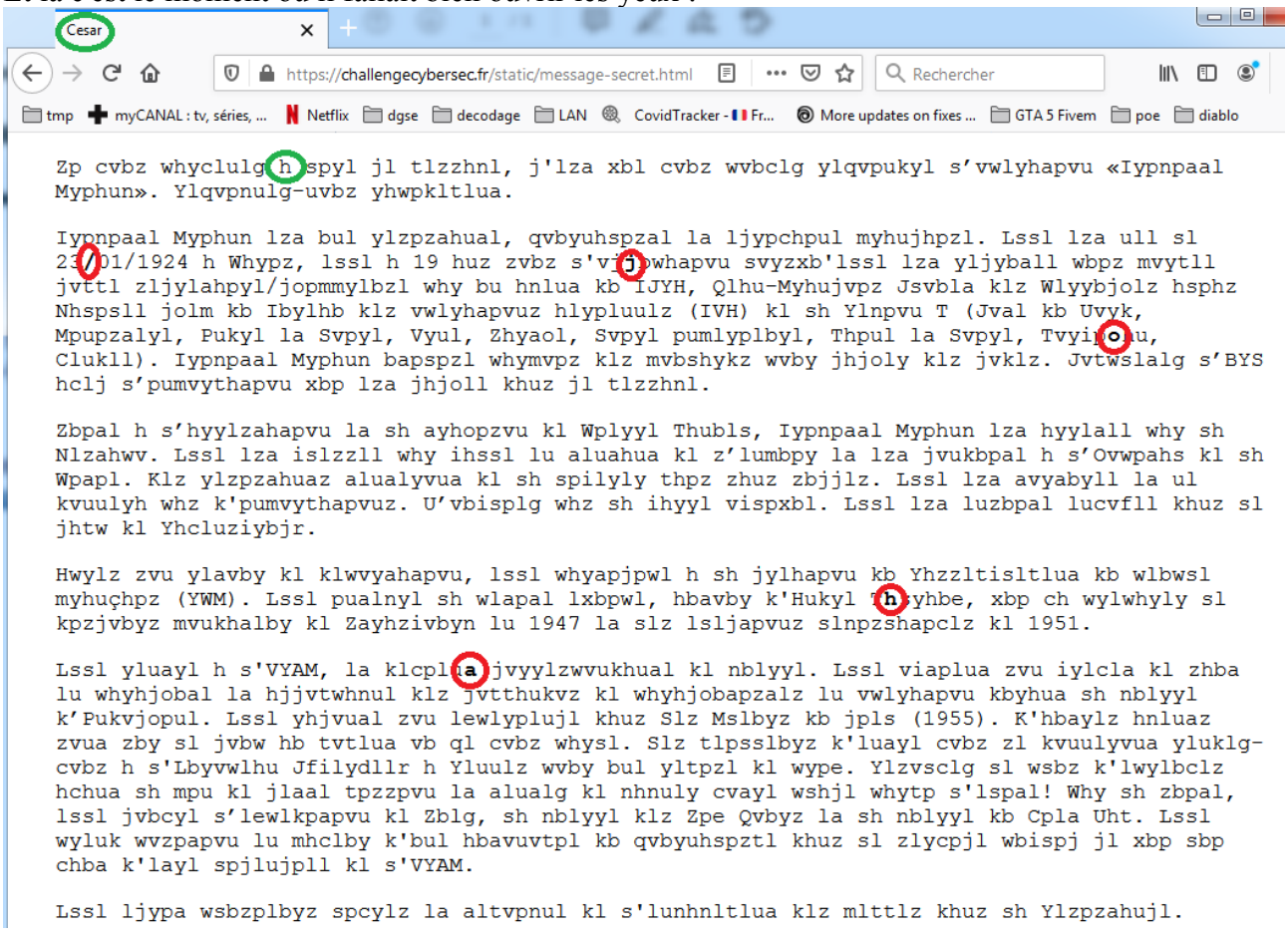
Le début du challenge

https://challengecybersec.fr/
A part un message indiquant de bien ouvrir les yeux, rien n'est visible sur la page.
En affichant le code source de la page on trouve une URL commentée.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Challenge Brigitte Friang</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <meta http-equiv="X-UA-Compatible" content="ie=edge" />
        <link rel="stylesheet" href="/static/bootstrap/css/bootstrap.min.css">
        <link rel="stylesheet" href="/static/css/style.css">
        <!--/static/message-secret.html-->
    </head>
    <body>
        <div class="container-effect">
            <div class="screen">
                <div id="particles-js"></div>
                <div class="h-100 d-flex justify-content-center align-items-center" id="test">
                    <div class="container">
```

On arrive donc sur la page
https://challengecybersec.fr/static/message-secret.html

Et la c'est le moment où il fallait bien ouvrir les yeux :



Zp cvbz whyclulg h spyl jl tlzzhnl, j'lza xbl cvbz wvbclg ylqvpukyl s'vwlyhapvu «Iypnpaal Myphun». Ylqvpnulg-uvbz yhwpkltlua.

Iypnpaal Myphun lza bul ylzpzahual, qvbyuhspzal la ljypchpul myhujhpzl. Lssl lza ull sl 23/01/1924 h Whypz, lssl h 19 huz zvbz s'vi whapvu svyzxb'lssl lza yljyball wbpz mvytll jvttl zljylahpyl/jopmmylbzl why bu hnlua kb IJYH, Qlhu-Myhujvpz Jsvbla klz Wlyybjolz hsphz Nhspsll jolm kb Ibylhb klz vwlyhapvuz hlypluulz (IVH) kl sh Ylnpvu T (Jval kb Uvyk, Mpupzalyl, Pukyl la Svpyl, Vyul, Zhyaol, Svpyl pumlyplbyl, Thpul la Svpyl, Tvyipolu, Clukll). Iypnpaal Myphun bapspzl whymvpz klz mvbshykz wvby jhjoly klz jvklz. Jvtwslalg s'BYS hclj s'pumvythapvu xbp lza jhjoll khuz jl tlzzhnl.

Zbpal h s'hyylzahapvu la sh ayhopzvu kl Wplyyl Thubls, Iypnpaal Myphun lza hyylall why sh Nlzahwv. Lssl lza islzzll why ihssl lu aluahua kl z'lumbpy la lza jvukbpal h s'Ovwpahs kl sh Wpapl. Klz ylzpzahuaz alualyvua kl sh spilyly thpz zhuz zbjjlz. Lssl lza avyabyll la ul kvuulyh whz k'pumvythapvuz. U'vbisplg whz sh ihyyl vispxbl. Lssl lza luzbpal lucvfll khuz sl jhtw kl Yhcluziybjr.

Hwylz zvu ylavby kl klwvyahapvu, lssl whyapjpwl h sh jylhapvu kb Yhzzltisltlua kb wlbwsl myhuçhpz (YWM). Lssl pualnyl sh wlapal lxbpwl, hbavby k'Hukyl h yhbe, xbp ch wylwhyly sl kpzjvbyz mvukhalby kl Zayhzivbyn lu 1947 la slz lsljapvuz slnpzshapclz kl 1951.

Lssl yluayl h s'VYAM, la klcpla jvyylzwvukhual kl nblyyl. Lssl viaplua zvu iylcla kl zhba lu whyhjobal la hjjvtwhnul klz jvtthukvz kl whyhjobapzalz lu vwlyhapvu kbyhua sh nblyyl k'Pukvjopul. Lssl yhjvual zvu lewlyplujl khuz Slz Mslbyz kb jpls (1955). K'hbaylz hnluaz zvua zby sl jvbw hb tvtlua vb ql cvbz whysl. Slz tlpsslbyz k'luayl cvbz zl kvuulyvua yluklg-cvbz h s'Lbyvwlhu Jfilydllr h Yluulz wvby bul yltpzl kl wype. Ylzvsclg sl wsbz k'lwylbclz hchua sh mpu kl jlaal tpzzpvu la alualg kl nhnuly cvayl wshjl whytp s'lspal! Why sh zbpal, lssl jvbcyl s'lewlkpapvu kl Zblg, sh nblyyl klz Zpe Qvbyz la sh nblyyl kb Cpla Uht. Lssl wyluk wvzpapvu lu mhclby k'bul hbavuvtpl kb qvbyuhspztl khuz sl zlycpjl wbispj jl xbp sbp chba k'layl spjlujpll kl s'VYAM.

Lssl ljypa wsbzplbyz spcylz la altvpnul kl s'lunhnltlua klz mlttlz khuz sh Ylzpzahujl.

On a un texte chiffré sous les yeux. La première chose à remarquer, est le titre de la page : Cesar. Le code Cesar consiste à décaler l'alphabet de n élements. n est donc la clé à trouver.
Entouré en vert, sur la capture d'écran on remarque un mot d'une seule lettre. Ici h chiffré, donc a si la clé est 7. (Il n'y a pas tellement de mots d'une seule lettre)
En utilisant 7 comme clé, on déchiffre bien le texte.

Si vous parvenez a lire ce message, c'est que vous pouvez rejoindre l'operation …..

On obtient un texte dans lequel certaines phrases nous indiquent qu'il faut trouver l'information cachée dans ce message.

Personellement, n'ayant pas suffisement ouvert les yeux, j'ai tourné en rond un petit moment avant de m'aperçevoir que le texte chiffré contenait les caractères /joha en gras.
Une fois déchiffrés, ceux-ci donne /chat

La suite se situe donc à l'adresse
https://challengecybersec.fr/chat

A cette adresse, on a une messagerie où le directeur Arman Richelieu vous invite à suivre 4 pistes pour découvrir les plans de l'Evil Gouv contre sa population de l'Evil country.

1ère piste : Antoine Rossignol – Service Crypto

Agent 42, félicitations pour votre affectation sur l'affaire Friang ! Je pense avoir une piste sérieuse. Mais avant toute chose, voici les échanges avec votre prédécesseur :

echange.txt

Eve Descartes est une spécialiste des attaques matérielles. Elle travaille dans les salles blanches d'ESIEE Paris et elle nous donne un coup de main de temps en temps. C'est la meilleure dans son domaine

Merci ! Je vous en dois une !

Attendez, je ne vous ai pas encore donné les fichiers !

archive_chiffree

layout.pdf

compte_rendu_eve.pdf

Comme vous pouvez le voir, le fichier que nous a envoyé Eve est protégé par un mot de passe et elle ne répond plus. Trouvez un moyen de la contacter pour déchiffrer le fichier

Décidément, vous ne chômez pas au service crypto !

Arrêtez votre charme 42. Nous cherchons une information capitale qui est certainement cachée dans archive_chiffree. Si vous trouvez quelque chose envoyez-moi un message !

Le fichier echange.txt

```
Antoine Rossignol 29/09/2020 : "Un des nos agents a intercepté du matériel de
chiffrement et un message chiffré émis par Evil Chems qui doit contenir des
informations capitales sur la livraison de produits chimiques."

Agent 40 30/09/2020 : "OK, on s'occupe de démonter le matériel pour analyse."
```

```
Agent 40 08/10/2020 : "C'est plus compliqué que prévu! Le processeur ne fait
qu'échanger des données avec un circuit intégré dédié (ASIC). On suppose qu'il
contient l'algorithme de chiffrement et que la clé y est stockée en dur."

Antoine Rossignol 08/10/2020 : " Envoyez en urgence l'ASIC à Eve Descartes
d'ESIEE-Paris pour une rétro-conception matérielle"

Agent 40 12/10/2020 : "Eve Descartes a bien reçu le circuit. Elle s'en occupe
en priorité."

Antoine Rossignol 23/10/2020 : "Voici le compte-rendu d'Eve avec la
cartographie de la zone étudiée. Il devient urgent de déchiffrer le message."
```

Le fichier archive_chiffree comme son nom l'indique est chiffré.
Le fichier layout.pdf est protégé par un mot de passe
seul le fichier compte_rendu_eve.pdf est accessible.

Ce compte rendu indique que dans le circuit intégré ont été trouvé 256 fusibles type Efuse contenant surement une clé, et que l'autre partie du circuit n'étant pas recouverte il s'agit sûrement de l'algo de chiffrement et d'un standard connu. Il indique que la photo des fusibles est dans le fichier layout.pdf

La 1ère chose qui nous est demandée est de trouver un moyen de contacter Eve.

ESIEE
PARIS

Service pour la Microélectronique et les Microsystèmes (S.M.M.)
ESIEE Paris - CCI en région Paris Ile-de-France
Cité Descartes / BP 99
2, Boulevard Blaise Pascal
93162 NOISY LE GRAND Cedex – France
http://www.esiee.fr/fr/ecole/salles-blanches

Ève Descartes
Ingénieur de Recherche Salle Blanche
Tél 01 45 92 60 96
E-mail. : eve.descartes@esiee.fr

A Noisy-le-Grand, le 23/10/2020

Objet : reverse circuit intégré

Monsieur,

Après avoir tenté tout et n'importe quoi, loin d'imaginer qu'un numéro de téléphone aurait été mis en place pour ce challenge, j'ai envoyé un mail à l'addresse email indiquée dans le document et obtenu un message d'absence :

absence Re: mdp layout.pdf Boîte de réception

DESCARTES, Eve<eve.descartes@esiee.fr>
À : eric.biton@gmail.com

Répondre | Répondre à tous | Transférer | Imprimer | Supprimer | Afficher l'original

Je suis actuellement indisponible.
Pour tout problème même minuscule, n'hésitez pas à me contacter par téléphone.

Si vous n'arriverez pas à me joindre par téléphone, c'est que la ligne est chargée. Réessayez ultérieurement.

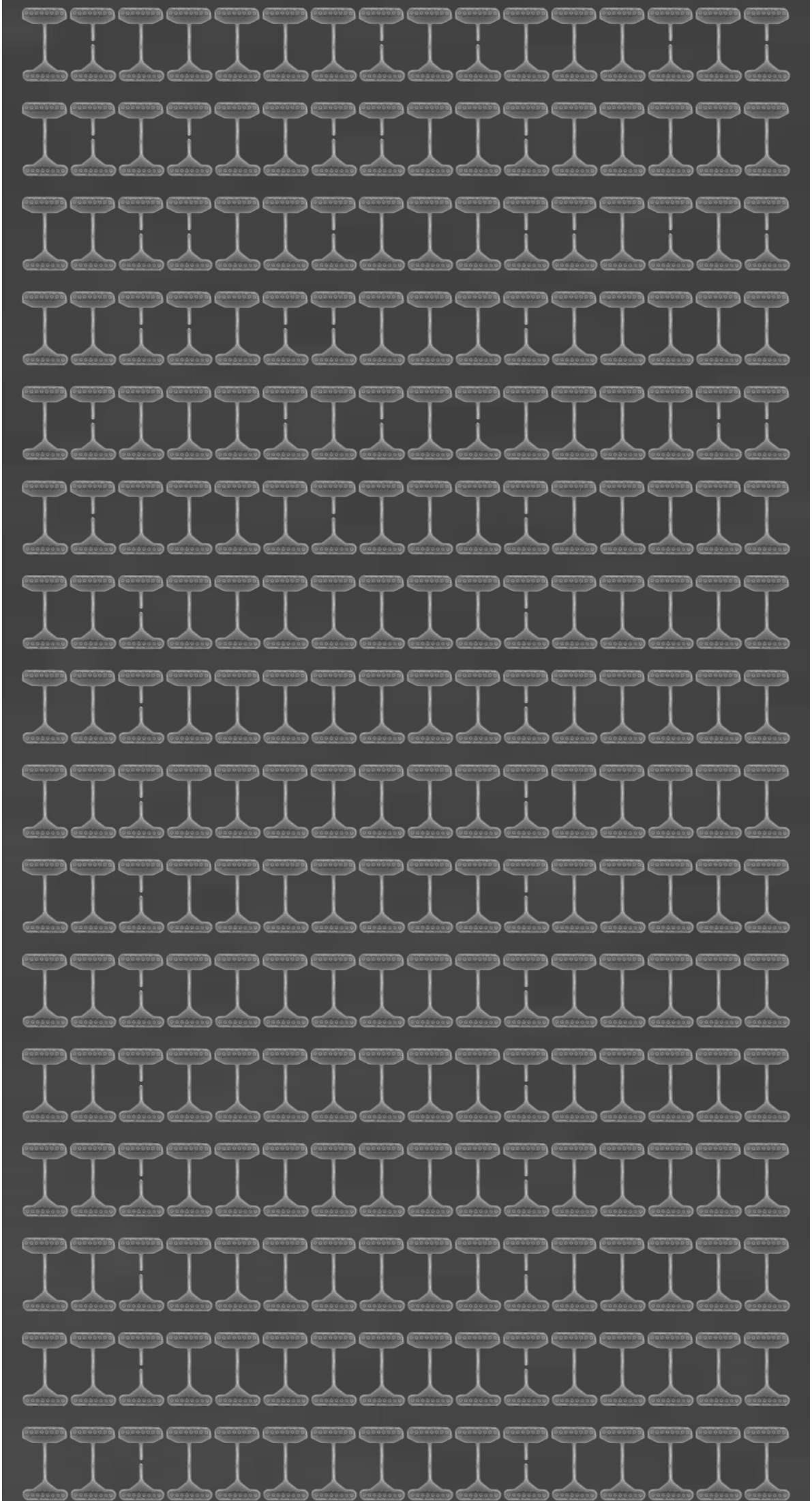
--
Eve Descartes
Ingénieur de Recherche Salle Blanche
Service pour la Microélectronique et les Microsystèmes (S.M.M.)
Tél. : +33 (0)1 45 92 60 96
2 boulevard Blaise Pascal - BP 99
93162 Noisy-le-Grand CEDEX • FRANCE
www.esiee.fr • www.univ-gustave-eiffel.fr

ESIEE PARIS — Membre fondateur Université Gustave Eiffel — une école de la CCI PARIS ILE-DE-FRANCE

J'ai donc insisté avec le numéro de téléphone et finit par avoir une série de bip sonores.

Ici avec un . Pour un bip court, un – pour un bip long et un espace pour les pauses voilà ce que j'ai entendu :

```
.-. . ... .. ... - .- -. -.-. .

Ce qui, déchiffré avec l'alphabet Morse, donne
.-. . ... .. ... - .- -. -.-. .
R   E S   I  S   T A  N   C   E

C'est donc le mot de passe du fichier layout.pdf (mais en
minuscules)

Le fichier pdf contient la photo suivante :
```



Je pars sur l'hypothèse que quand il n'y a pas de coupure, c'est un bit à 1, et quand il y en a c'est un bit à 0 et note chacun des 256 bits.

Après plusieurs tentatives ratées de faire quelque chose avec cette suite de bits, je me dis qu'à tous les coups j'ai inversé les 1 et 0.
Je m'aperçois à l'aide d'un éditeur Hexadécimal, que cette suite donne la chaîne «AES 256 ECB «

on utilise open ssl avec l'algo AES longueur de clé 256 et mode opératoire ECB pour déchiffrer le fichier archive_chifree
avec la commande suivante (ici la clé est entrée en hexadécimal)

```
openssl aes-256-ecb -d -K
41455320323536204543422020202020202020202020202020202020202020
-in d:\archive_chiffree.txt -out archive_dechiffree1.txt
```

(Petite parenthèse : le mode ECB chiffre indépendament chaque bloc. Cela à l'avange de pouvoir paralleliser le chiffrement, mais l'inconvénient est qu'un bloc chiffré peut être remplacé par un autre bloc chiffré sans que cela ne fasse échouer le déchiffrement : sensible aux attaques par dictionnaire)

J'ouvre le fichier archive_dechiffree1.txt avec un editeur de texte. Illisible mais commence par PK ==> c'est un fichier zip

Je renomme le fichier en fichier zip, et dans ces fichiers on découvre 2 pdf (dédoublés pour différents lecteurs de ce format j'imagine)

On a donc un fichier code_acces.pdf protégé par un mot de passe et un fichier message.pdf lisible.
Celui ci indique que l'autre fichier est protégé par un mot de passe x avec :

$$\text{Le mot de passe est } x \text{ tel que } \begin{cases} x^3 \equiv 573 \left[8387 \right] \\ x^3 \equiv 98 \left[9613 \right] \\ x^3 \equiv 2726 \left[7927 \right] \end{cases}$$

Ma première réaction fut : c'est quoi ces = avec 3 barres !!!
Ca vous donne idée de mon niveau en maths :-)

Après avoir visité quelques pages Wikipédia, je comprends qu'il s'agit de congruences et que ceci signifie que le reste de la divion de x au cube par 8387 est 573,le reste de la divion de x au cube par 9613 est 98 et le reste de la divion de x au cube par

7927 est 2726

Ne sachant comme résoudre ces congruences, voilà comment j'ai procédé :
Pour commencer je pose A = x^3
selon la 2ème ligne le reste de A divisé par 9613 est 98
J'ai donc un entier i tel que A = 9613*i + 98
donc A = 9613*i +98
en utilisant la 1ère ligne le reste de A divisé par 8387 est 573
donc pour cette ligne A – 573 est divible par 8387

pour trouver i qui vérifie ces 2 lignes je cherche i qui vérifie que 9613*i + 98 – 573 est divisible par 8387 donc 9613*i – 475 | 8387

une fois que j'ai un i qui vérifie les 2, il faut vérifier que la 3ème ligne est vérifiée.
Et donc que 9613*i + 98 - 2726 est divisible par 7927 donc 9613*i – 2628 | 7927

Bref j'ai écris vite fait le code php suivant :

```
<!DOCTYPE html>
<html><body><pre>
<?php
for($i=1;$i<500000000;$i++){
  $s = (($i * 9613) - 475)/8387;
  if(round($s,0)==$s){
    $t = (($i * 9613) -2628)/7927;
    if (round($t,0)==$t){
      echo $i."\n";
    }
  }
}
?>
</pre></body></html>
```

Le premier résultat est i=18484750 , je le multiplie par 9613, lui ajoute 98 pour avoir A ,j'en fais la racine cubique pour avoir x : j'obtiens 5622

C'est le mot de passe du fichier code_acces.pdf

(Après en avoir lu un peu plus sur les congruences, on trouve A en utilisant le théorème des restes chinois : mais bon je suis visiblement plus codeur que matheux. Et pour le coup en revoyant je me demande pourquoi j'ai utilisé le round et pas testé le modulo ? … mystère !)

On arrive à ouvrir le fichier code_acces.pdf et la on nous explique que le mot de passe chiffré est 0xAF3A5E20A63AD0
et que chaque caractère du mot de passe a été remplacé par son inverse dans GF(256) = $Z_2[X]/(X^8+X^4+X^3+X+1)$

Et la, après avoir pas mal lu sur AES, j'ai fait le rapprochement entre AES et le polynôme $X^8+X^4+X^3+X+1$

Quelques recherches google plus tard je suis tombé sur cette page http://www.cs.utsa.edu/~wagner/laws/FFM.html
qui contient à la fin une table des inverses.

Donc pour AF, le premier caractère du mot de passe chiffré, en lisant la ligne a, et la colonne f j'obtiens 62 le code hexa du premiere caractère.
Ce qui donne 62 20 61 3A 65 20 7A en hexadécimal donc la chaîne b a:e z

On la tape dans le chat avec Antoine Rossignol qui nous réponds :

C'est ça ! Tiens, ça me fait penser à Enigma. Notez-le ça pourrait vous servir ultérieurement... On sait maintenant qu'Evil Country va utiliser du VX contre sa propre population, connectez-vous sur cette plateforme top secrète pour continuer l'enquête : /7a144cdc500b28e80cf760d60aca2ed3, on ne peut pas prendre le risque qu'un agent double ne s'en rende compte.

Donc la suite se trouve à l'URL
https://challengecybersec.fr/7a144cdc500b28e80cf760d60aca2ed3

Et c'est la Fin de la 1ère piste

2ème piste Jérémy Nitel - Service Web

Jérémy vous demande de retrouver sur la plateforme web Stockos d'un entrepôt l'adresse email d'un client qui utilise des marchandises lié au massacre des résistants.

Il précise que Stockos a connu plusieurs fuites de mots de passe et qu'ils étaient pas très originaux.
Il donne le lien vers Stockos

https://challengecybersec.fr/4e9033c6eacf38dc2a5df7a14526bec1

Enfin il demande, grâce aux infos obtenues, de réserver un vol sur le site AirEvil dont il donne le lien.

On se connecte à Stockos. On cherche un identifiant et un user pas très originaux : admin admin

Une fois connecté sur le site il n'y a finalement qu'une page avec laquelle on peut intéragir : la page de gestion des stocks où on peut rechercher un objet.

On voit d'ailleurs que le 1er objet de la liste est du cyclosarin donc ca doit être le client qui a commandé cela dont on doit retrouver l'email.
En entrant uniquement le caractère ' dans le champ de recherche on obtient

```
Erreur : You have an error in your SQL syntax; check the manual
that corresponds to your MySQL server version for the right syntax
to use near '%' ORDER BY section ASC' at line 1
```

On en déduit que le site exécute un ordre sql qui se termine par `LIKE '%[XXXX]%'` ORDER BY section ASC avec [XXXX] ce qu'on a tapé dans le champ de recherche sans avoir protégé la requête contre l'injection de sql.

En entrant juste ' on a « fermé » le like, ce qu'on tape derrière ce ' se retrouve donc l'ordre sql exécuté.
On va utiliser ici UNION pour ajouter aux résultats les résultats d'un ordre select qu'on aura entré en finissant ce qu'on entre dans le champ par deux tirets et un espace afin que le reste soit en commentaire pour mysql.
Dans la requête insérée avec union, on donnera des alias au noms de colonnes pour que les données soient ajoutées au tableau sur la page web.

Soit on devine qu'il y a une table customer avec une colonne email (c'est comme ca que j'ai trouvé), soit on fait des requêtes sur les tables INFORMATION_SCHEMA pour connaître les tables et colonnes.
Au final en entrant dans la recherche ceci

```
' union select id, email object,1 status,2 section, 3 client from customer --
```

A la fin du tableau on été ajoutés des lignes avec l'email des client dans la colonne objet.
On tombe sur l'adresse mail agent.malice@secret.evil.gov.ev

Il faut maintenant passer au site d'AirEvil.
Ce site utilise l'email comme identifiant pour la connexion. On peut se créer un compte sur le site.
On reçoit un email pour activer le compte
Mais avec notre compte créé on ne peut pas réserver le vol demandé.
Nous connaissons l'email de l'agent malice, mais n'avons pas son mot de passe.

Sous le formulaire de connexion, il y a lien pour le mot de passe oublié.
La il faut entrer son adresse mail et un mail est envoyé. On a pas accès à la messagerie de l'agent malice donc cela nous sert à rien, mais on peut tester la procédure avec son propre compte.

Comme pour le mail d'activation, le mail pour retrouver son mot de passe contient un lien.
On remarque que pour les 2 mails, le lien se termine pareil :

Air-Evil - Activation compte Boîte de réception

air-evil@challengecybersec.fr<air-evil@challengecybersec.fr>
À : eric.biton@gmail.com

Répondre | Répondre à tous | Transférer | Imprimer | Supprimer | Afficher l'original

Prêt à voyager au pays des merveilles ?

Activez vôtre compte dès maintenant !

http://challengecybersec.fr/35e334a1ef338faf064da9eb5f861d3c/activate/ZXJpYy5iaXRvbkBnbWFpbC5jb20=

Vous avez jusqu'à Sat Oct 31 2020 18:58:28 GMT+0000 (Coordinated Universal Time) le faire

Air-Evil - Changement de mot de passe Boîte de réception

air-evil@challengecybersec.fr<air-evil@challengecybersec.fr>
À : eric.biton@gmail.com

Répondre | Répondre à tous | Transférer | Imprimer | Supprimer | Afficher l'original

Vous recevez ce message car vous (ou quelqu'un d'autre) a demandé la reinitialisation de vore mot de passe

Merci de clicker sur le lien suivant ou de le coller dans votre navigateur pour terminer le processus de reinitialisation

http://challengecybersec.fr/35e334a1ef338faf064da9eb5f861d3c/reset/ZXJpYy5iaXRvbkBnbWFpbC5jb20=

Vous avez jusqu'à Invalid Date pour modifier votre mot de passe

Si vous n'avez pas demandé de reinitialisation, ignorez ce message

Répondre | Répondre à tous | Transférer | Imprimer | Supprimer | Afficher l'original

La chaîne de caractères derrière activate ou reset (et qui se termine par =), ça ressemble à du texte encodé en base64.

Je décode donc le ZXJpYy5iaXRvbkBnbWFpbC5jb20= avec base64 ca me donne mon adresse mail.

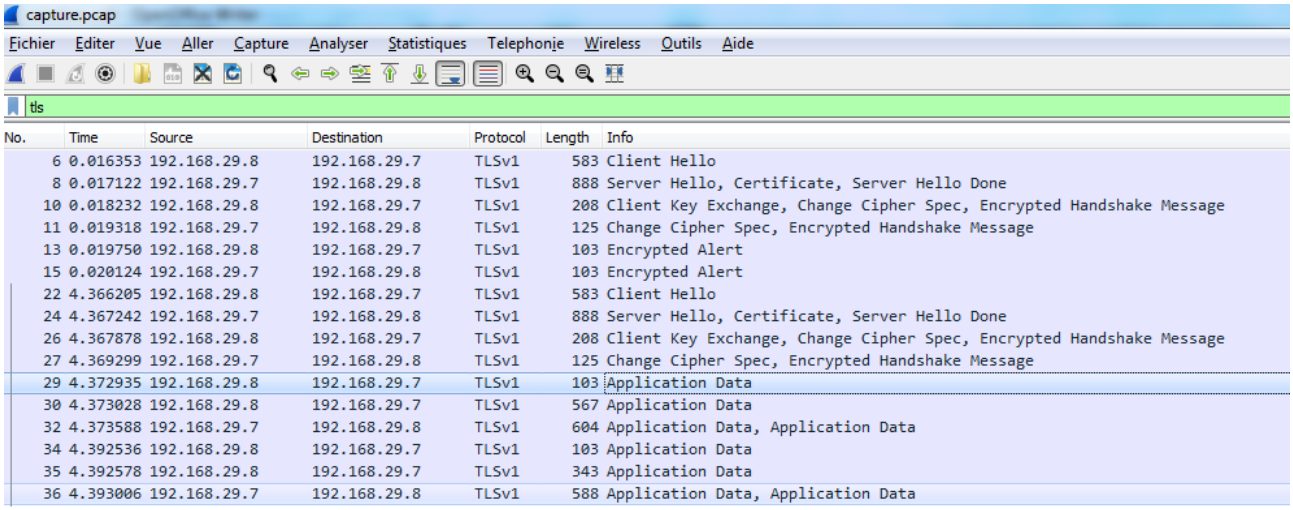
==> j'encode en base64 l'email de l'agent malice et je le remplace dans l'URL
==> je me fais jeter avec un message jeton invalide (c'est parce qu'on a pas demandé la « réinitialisation » pour lui). Je demande la réinitialisation avec son email puis j'utilise l'adresse …/reset/.... avec son email encodé en base64
Le site me donne le mot de passe du compte : Superlongpassword666

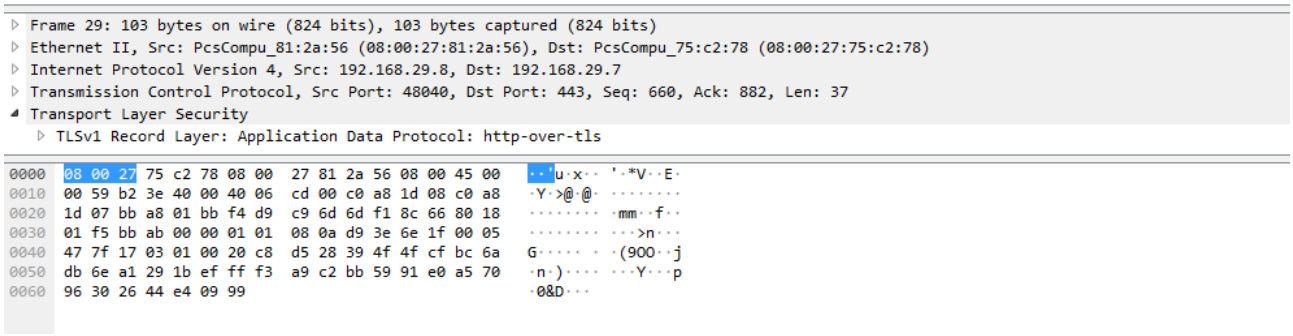
On peut donc se connecter au site AirEvil avec ces identifiants et commander le vol si ce n'est déjà fait (moi c'était déjà fait donc dans « Mes réservations » on retrouve le billet)

On scanne le Qrcode, et on obtient le flag
DGSESIEE{2cd992f9b2319860ce3a35db6673a9b8}qu'on tape dans le chat avec Jéremy Nitel

Il vous donne alors un fichier sur une communication interceptée.
Ce fichier capture.pcap s'ouvre avec WireShark.
On voit qu'il s'agit d'un échange en TLS 1 donc chiffré.

Ce qui va nous intéresser c'est le contenu des trames Application data

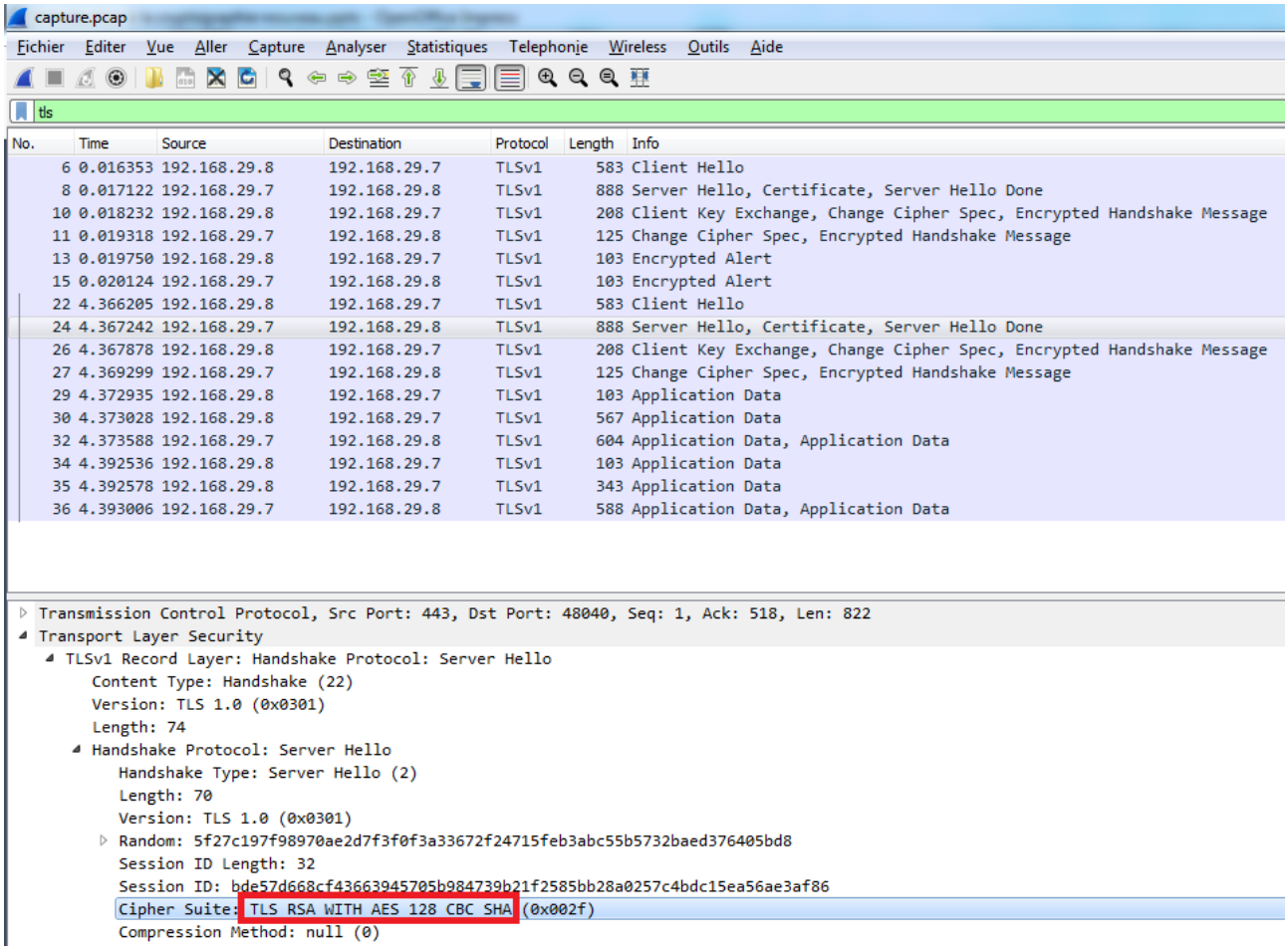




Mais la elles ne sont pas visibles.

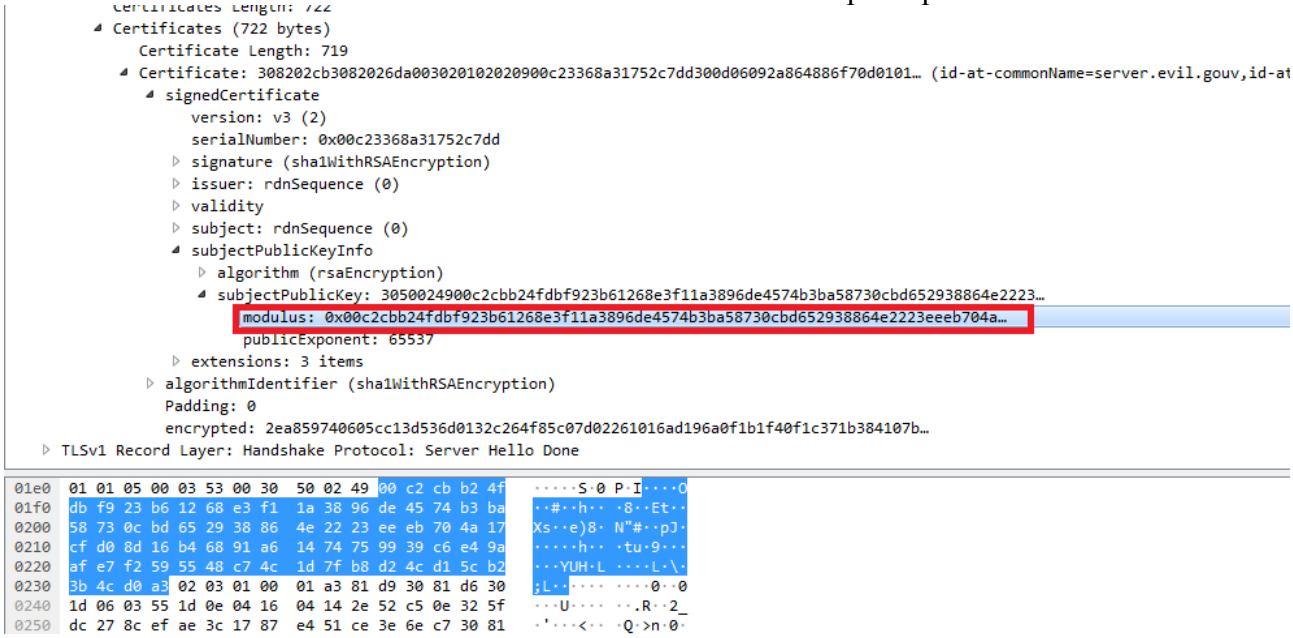
Il faut donc d'abord savoir comment marche globalement TLS :
En simplifié :
Le client envoie un Client Hello au server comprenant notamment les suites de chifffrement qu'il sait gérer
Le serveur envoie un Server Hello avec la suite de chiffrement choisie et la clé publique du serveur.
Le client envoie un Client Key Exchange contenant la clé de chiffrement qui sera utilisée pour la suite de la communication, cette clé elle même chiffrée avec la clé publique du serveur. (Donc uniquement déchiffrable avec la clé privée du serveur donc uniquement par le serveur).

On regarde le Server Hello :



On voit le chiffrement choisi par le serveur Donc RSA pour le chifffrement de la clé, puis AEC 128 en mode CBC pour la suite de la communication.

Dans la même trame on trouve le certificat du serveur avec sa clé publique



On remarque que le modulus (produit des 2 grands nombres premiers à l'origine de la clé privée) est «très petit»

Si on connait RSA (sinon la page wikipédia l'explique très bien), on sait que casser le RSA, c'est réussir à factoriser le modulus en p et q tous 2 nombres premiers.
(Générez toujours des clés privées de 2048 bits : au dela c'est de l'overkill, et même en 8192 bits,

l'informatique quantique la cassera, à ce moment la il faudra utiliser la crypto post-quantique)

Si on arrive à trouver ces 2 nombres premiers , on peut calculer tous les éléments nécessaires pour recréer la clé privée et ainsi déchiffrer le Client Key Exchange.

Ce topic sur stack overflow détaille la structure du fichier à générer et les commandes openssl pour faire cela.
https://stackoverflow.com/questions/19850283/how-to-generate-rsa-keys-using-specific-input-numbers-in-openssl

Pour factoriser, il existe un site http://factordb.com/

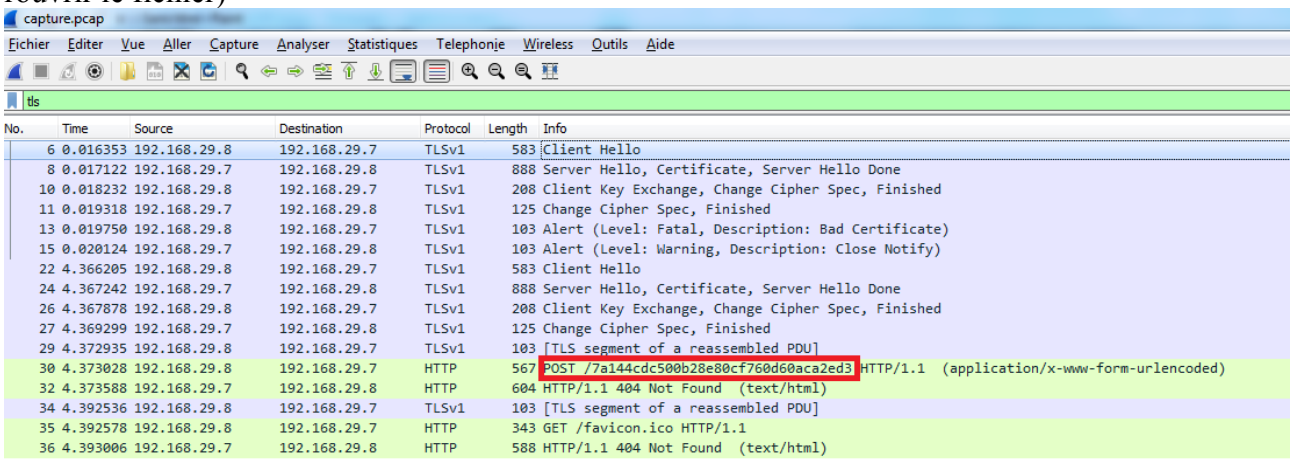
sur lequel vous pouvez entrer le modulus (converti en décimal)
18819881292060796383869723946165043980716356337941738270076335642298885971523466548531906060650474304531738801130339671619969232120573403187955065699622130516875930765025705 9

qui va vous donner les 2 nombres
39807508642406493739712550055038649119906436234252670840638518957594638895726176858331 7
et
47277214610743530253622307197304822463291469530209711645985217113052071125636359039752 7

Vous pouvez alors calculer les exposants servant à refaire la clé privée.

Et si vous avez pas envie de faire tout ça à la main, vous pouvez utiliser un script python que vous trouverez sur Github qui s'appelle RsaCtfTool qui peut prendre en entrée les premiers p et q ou directement le modulus (il utilise d'ailleurs factordb.com aussi) ou le certificat... et vous génère la clé privée.

Une fois la clé privée générée, vous pouver l'importer dans WireShark (Editer->Préférences->RsaKey)
Une fois la clé importée, Wireshark l'utilisera pour déchiffrer tout seul les trames (mais il faut rouvrir le fichier)



La on voit une requête POST sur /7a144cdc500b28e80cf760d60aca2ed3

La suite à l'URL
https://challengecybersec.fr/7a144cdc500b28e80cf760d60aca2ed3

Fin de la piste Web.

3ème piste Blaise Pascal – Service Algo

Blaise Pascal nous donne 2 fichiers. Un fichier texte original et un fichier intercepté dont les hash ne correspondent pas et demande de retrouver l'information cachée.

En regardant les 2 fichiers texte, on s'aperçoit que le fichier intercepté est le fichier original avec des caractères ajoutés.

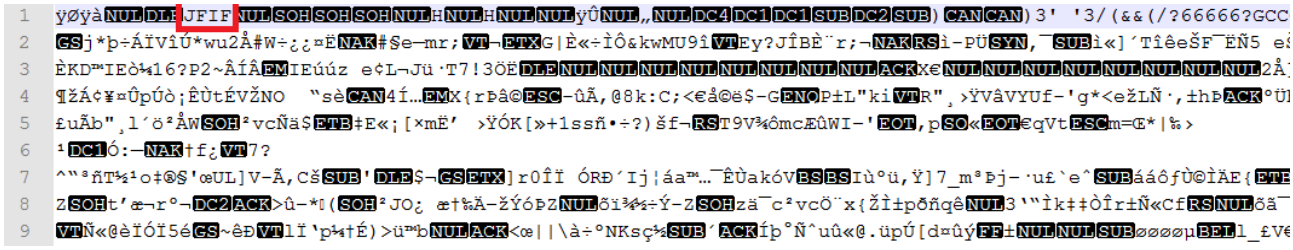
Je fais un petit code php pour afficher les caractères ajoutés

```
<!DOCTYPE html>
<html><body><pre>
<?php
$hO = fopen("D:\\challenge\\blaise pascal\\original.txt","r");
$hI = fopen("D:\\challenge\\blaise pascal\\intercepte.txt","r");

$b="";
$cO = fgetc($hO);
while (false !== ($cI = fgetc($hI))) {
  if($cI!=$cO){
    $b.=$cI;
  } else {
    $cO = fgetc($hO);
  }
}

fclose($hO);
fclose($hI);
//echo $b;
file_put_contents("D:\\challenge\\blaise
pascal\\hidden.txt",base64_decode(substr($b,7)));
?>
</pre></body></html>
```

Après avoir afficher les caractères ajoutés, je vois que le résultat est «base64:.....»

Je commente le echo, décode le base64 et l'enregistre dans un fichier.

J'ouvre le fichier avec un éditeur de texte


ÿØÿà NUL DLE JFIF NUL SOH SOH SOH NUL H NUL H NUL NUL ýÛ NUL „ NUL DC4 DC1 DC1 SUB DC2 SUB) CAN CAN) 3' '3/(&&(/?66666?GCC
GS j*þ÷ÁÏVíÚ*wu2Å#W÷¿¿¤ÉNAK#$e—mr;VT¬ETXG|È«÷ÌÔ&kwMU9îVTEy?JÎBÈ¨r;¬NAKRSì-PÜSYN,¯SUBi«]´TîêeŠF¯ËÑ5 eš
ÈKD™IEò¼16?P2~ÂÍÂEMIEúúz e¢L¬Jü·T7!3ÖËDLE NUL NUL NUL NUL NUL NUL NUL NUL ACK X€ NUL NUL NUL NUL NUL NUL NUL NUL 2Å
¶ŽÁ¢¥¤ÛpÚò¡ÊÙtÉVŽNO "sèCAN4Í…EMX{rÞâ©ESC-ûÃ,@8k:C;<€å©ë$-GENOP±L"kiVTR"¸›ŸVâVYUf-'g*<eŽLÑ·,±hÞACK°Ü
£uÃb"¸l´ö²ÅWSOH²vcÑä$ETB‡E«¡[×mË' ›ŸÓK[»+1ssñ•÷?)šf¬RST9V¾ômcÆûWI-'EOT,pSO«EOT€qVtESCm=Œ*|‰›
¹DC1Ó:—NAK†f¿VT7?
^"³ñT½¹o‡®§'œUL]V-Ã,CšSUB'DLE$¬GSETX]r0ÎÏ ÓRÐ´Ij¦áa™…¯ÊÙakóVBSBSIù°ü,Ÿ]7_m³Þj-·u£`e^SUBáâôƒÙ©ÌÄE{ETB
ZSOHt'æ¬r°¬DC2ACK>û-*▯(SOH²JO¿ æ†‰Ä-ŽÝóÞZNULõï¾½÷Ý-ZSOHzä¯c²vcÖ¨x{ŽÌ±pðñqêNUL3'"Ìk‡‡ÒÎr±Ñ«CfRSNULõã¯
VTÑ«@èÏÓÏ5éGS~êÐVTlÏ'p¼†É)>ü™bNULACK<œ||\à÷°NKsç½SUB´ACKÍþ°Ñ^uû«@.üpÚ[d¤ûýFF±NULNULSUBøøøøµBELl_£V€
La tête du début de fichier ma fait penser à une image jpeg
Je renomme le fichier en .jpg

On a /22CAEEE05CB8B2A49133BE134A5E9432

la suite à l'url
https://challengecybersec.fr/22caeee05cb8b2a49133be134a5e9432

Sur ce site on trouve une archive à télécharger contenant des instructions dans un PDF

Le pdf explique que l'on a 4 fichiers .in
constitué de 2 lignes
sur la première on a une valeur à atteindre et un nombre d'objets
sur la 2ème ligne on a la valeur des différents objets

Le but est de choisir les objets de la liste à stocker pour atteindre la valeur, sachant qu'un objet ne peut être stockée qu'1 fois et la valeur totale stockée ne peut dépasser la valeur à atteindre.

Il explique comment doit être exprimée la solution: 1 fichier de 2 lignes avec en première ligne le nombre d'objets et en 2ème lignes les index des objets à mettre en stock.

Bien entendu il y a un fichier avec une liste de 15000 objets, donc le faire à la main c'est impensable.

Pour ce genre de problème d'optimisation d'occupation de place, on devrait sûrement trouver des algos tout fait, mais c'est pas super drôle donc voila ma solution, peut être imparfaite, mais qui pour chacun des 4 fichiers à donné la solution atteignant pile poile la valeur à atteindre.

En gros, et en français:

Je commence par trier les objets par valeur.

Dans une var bestSolution, je stocke les objets jusqu'à ce qu'il n'y ait plus de place.

Pour chaque objet non pris, je regarde si je peux enlever un objet pris pour ajouter celui la.
Si au cours de la boucle j'ai modifié quelque chose, on recommence

Pour chaque objet non pris, je regarde s'il y a moyen d'enlever n objets pris pour ajouter celui la : pour cela j'appelle récursivement ma fonction d'optimisation mais ce coup ci sur les objets déjà pris en stock, avec valeur max à atteindre = la différence entre la valeur actuelle du stock + celle de l'objet que je veux ajouter et la valeur max à atteindre pour le stock.
Si au cours de la boucle j'ai modifié quelque chose, on recommence

le code php :

```
<!DOCTYPE html>
<html><body><pre>
<?php
ini_set('max_execution_time',120);
class solution{
  var $nbObj;
  var $val;
  var $maxVal;
  var $objects;

  public function __construct($maxval=0){
    $this->nbObj=0;
    $this->val=0;
    $this->maxVal=$maxval;
    $this->objects = array();
  }

  public function __toString(){
    $r=$this->nbObj."\n";
    $r.= implode(" ",array_keys($this->objects));
    return $r;
  }

  //ajout un objet retourne false si pas possible
  public function addObj($val,$index){
    if(isset($this->objects[$index])) return false;
    if($this->val + $val > $this->maxVal) return false;
    $this->objects[$index] = $val;
    $this->val+=$val;
    $this->nbObj++;
    return true;
  }

  public function inStock($index){
    return isset($this->objects[$index]);
  }

  public function remove($index){
    $this->nbObj--;
    $this->val-=$this->objects[$index];
    unset($this->objects[$index]);
  }
}

//$filename="exemple";
//$filename="fichier_a_petit";
//$filename="fichier_b_moyen";
//$filename="fichier_c_gros";
$filename="fichier_d_gros";

function optimizeStock($l,$mv){
  if(is_object($l)){ //sert à rien la mais bon du à mes multiples modifs
    $liste= clone $l;
  } else {
    $liste=$l;
  }

  if(is_object($mv)){ //sert à rien la mais bon du à mes multiples modifs
    $maxVal= clone $mv;
  } else {
    $maxVal=$mv;
  }
```

```
  asort($liste); // trie en conservant les index
  $bestSolution = new solution($maxVal);
  $nb = count($liste);

  //bourre les plus petits objets jusqu'à plus de place
  if(current($liste)!==false){
    while($bestSolution->addObj(current($liste),key($liste))){
      if(!next($liste)) break;
    }
  }

  //Tente le remplacement de 1 pour 1 autre
  $encore3 = true;
  for($passe=0;$encore3;$passe++){
    $encore3=false;
    reset($liste);
    $encore2=true;
    while($encore2 &&($bestSolution->val < $bestSolution->maxVal)){
      if(!$bestSolution->inStock(key($liste))){
        $diff = current($liste) + $bestSolution->val - $maxVal;
        asort($bestSolution->objects);
        reset($bestSolution->objects);
        $end=false;
        $encore=true;
        $v=-1;
        while(!$end && $encore){
          $v=current($bestSolution->objects);
          if($v < $diff){
            $encore=next($bestSolution->objects);
          } else {
            $end=true;
            if(($maxVal - $bestSolution->val) > ($maxVal - ($bestSolution->val
- current($bestSolution->objects) + current($liste)))){
              $bestSolution->remove(key($bestSolution->objects));
              $bestSolution->addObj(current($liste),key($liste));
              $encore3=true;
            }
          }
        }
      }
      $encore2 = next($liste);
    }
  }

  //Tente le remplacement de n par 1
  $encore3 = true;
  for($passe=0;$encore3;$passe++){
    $encore3=false;
    reset($liste);
    $encore2=true;
    while($encore2 &&($bestSolution->val < $bestSolution->maxVal)){
      if(!$bestSolution->inStock(key($liste))){
        $toRemove = current($liste) - $bestSolution->maxVal + $bestSolution-
>val;
        if($toRemove > 0){
          $rmvSolution = optimizeStock($bestSolution->objects,$toRemove);
          if($rmvSolution->nbObj > 0){
            $encore3=true;
            foreach($rmvSolution->objects as $k => $v){
              $bestSolution->remove($k);
            }
            $bestSolution->addObj(current($liste),key($liste));
          }
```

```
        }
      }
      $encore2 = next($liste);
    }
  }

  ksort($bestSolution->objects); //retrie par clé comme ça on peut "echo" la
solution direct
  return $bestSolution;
}


$f = file_get_contents("D:\\challenge\\blaise pascal\\archive\\".
$filename.".in");
$linesI = explode("\n",$f);

$maxVal = explode(" ",$linesI[0])[0];
$objets=explode(" ",$linesI[1]);

$bestSolution = optimizeStock($objets,$maxVal);

file_put_contents("D:\\challenge\\blaise pascal\\archive\\".$filename.".txt",
$bestSolution);
?>
</pre></body></html>
```

une fois la solution validée, on nous donne l'adresse de l'étape suivante :
https://challengecybersec.fr/9bcb53d26eab7e9e08cc9ffae4396b48

La on nous indique que ce blog contient un grand nombre de posts. Chacun ayant un message-digest, et nous demande de concatener tous ces hash puis d'en faire le md5.

Sur la droite, on trouve des liens vers d'autres articles
On constate en visitant les pages de ces articles que l'url se termine par /post/[9999] avec [9999] l'id du post
chaque post se termine par le message-digest affiché.

En remplacant l'id dans l'url on voit qu'il y a mille posts.

Je suis sous windows et j'ai curl et grep d'installés : je fais un petit .bat

```
curl -k --ssl-no-revoke
https://challengecybersec.fr/9bcb53d26eab7e9e08cc9ffae4396b48/blog/post/1    |
grep digest > digest.txt
FOR /L %%x IN (2,1,1000) DO (
  echo %%x>> digest.txt
  curl -k --ssl-no-revoke
https://challengecybersec.fr/9bcb53d26eab7e9e08cc9ffae4396b48/blog/post/%%x
| grep digest >> digest.txt
)
```

Ici j'utilise l'option -ssl-no-revoke histoire de pas checker mille fois la révocation du certificat ce qui prends du temps et je suis pressé (mais c'est mal!)

J'obtiens donc un fichier texte rempli de 1000 lignes de ce type
message-digest: <span id="partial-
proof">3a59847fb7d7e4b6c02d88f01a17c66f

La je me suis pas embêté : mode colonnes de notepad++ pour garder que les hash, puis un petit remplacer pour enlever les retour à la ligne et l'option outil->md5 sur le tout

on se retrouve à l'étape suivante:
https://challengecybersec.fr/1410e53b7550c466c76fc7268a8160ae

ici on a un formulaire de login. Si on regarde les sources on voit que lorsqu'on valide l'utilisateur (et non le mot de passe) est passé à script javascript de validation, et s'il est valide alors on est redirigé vers l'url actuelle + / + l'utilisateur entré.

La code se trouve dans un fichier login.js dont le code est quasi illisible sur une seule ligne :

```
var
_0x5f46=['\x37\x3c\x30\x6c\x3c\x6e\x69\x30\x33\x3c\x6c\x3c\x6c\x3c\x33\x3e\x35\x3c\x62\x60\x3e\x64\
x6b\x3e\x6a\x3b\x33\x6e\x30\x3e\x3e\x6f\x39\x6e\x30\x60\x6e\x6b\x33\x39','\x39\x6f\x23\x6a\x7a\x51\
x24\x3d\x57\x38\x73\x4e\x3e\x6e\x3f\x6b\x49\x58\x75\x49\x4d\x37\x73\x68\x36\x20\x57\x69\x6c\x62\x44
\x50\x78\x60\x31\x26\x59\x46\x35\x7a','\x6c\x65\x6e\x67\x74\x68','\x73\x70\x6c\x69\x74','\x63\x68\x
61\x72\x41\x74','\x6a\x6f\x69\x6e','\x64\x5b\x63\x6e\x3f\x6b\x2b\x71\x6a\x65\x29\x2f\x4e\x7c\x74\x2
e\x77\x6b\x47\x72\x5d\x72\x4f\x2b\x6b\x39\x62\x3d\x32\x79\x2c\x7d\x40\x5a\x79\x62\x3a\x38\x70\x6c\x
61\x32\x27\x36\x25\x64\x6e\x29','\x63\x68\x61\x72\x43\x6f\x64\x65\x41\x74','\x66\x72\x6f\x6d\x43\x6
8\x61\x72\x43\x6f\x64\x65'];var _0x19fd=function(_0x5f4656,_0x19fd1f){_0x5f4656=_0x5f4656-0x0;var
_0x79ddde=_0x5f46[_0x5f4656];return _0x79ddde;};function _0x10dbec(_0x1975bf){var
_0x1b7fa0=_0x19fd('\x30\x78\x30');var _0x4e7c63=0x0;var
_0x6ae845=_0x4bf1ad(_0x53e54e(_0x1975bf));if(_0x6ae845==_0x1b7fa0)
{_0x4e7c63=0x1;}else{_0x4e7c63=0x0;}return _0x4e7c63;}function _0x44d925(){var
_0x44809b=[0x2,0x15,0x0,0x22,0xb,0x9,0x17,0x1e,0xe,0x5,0x1d,0x4,0x18,0x16,0x8,0x14,0x1f,0x11,0x26,0
x23,0xf,0x1,0xd,0x6,0xc,0x1a,0x19,0x1b,0x21,0xa,0x7,0x10,0x20,0x1c,0x3,0x13,0x25,0x24,0x12,0x27];re
turn _0x44809b;}function _0x22f9d2(){var
_0xb974a1=[0x0,0x15,0x0,0x22,0x4,0x9,0x17,0x1e,0xe,0x5,0x1d,0x4,0x18,0x16,0x8,0x14,0x1f,0x11,0x26,0
x23,0xf,0x1,0xd,0x6,0xc,0x1a,0x19,0x1b,0x21,0xa,0x7,0x10,0x20,0x1c,0x3,0x13,0x25,0x24,0x12,0x27];re
turn _0xb974a1;}function _0xdbb8b3(){var
_0x22dcfa=[0x0,0x15,0x0,0x22,0x4,0x9,0x17,0x7,0xe,0x5,0x1d,0x4,0x18,0xd,0x8,0x14,0x1f,0x11,0x26,0x2
3,0xf,0x1,0xd,0x6,0xc,0x1a,0x19,0x1b,0x21,0xa,0x7,0x10,0x20,0x1c,0x3,0x13,0x25,0x24,0x12,0x27];retu
rn _0x22dcfa;}function _0x33903e(_0x3c3da1){var _0x334fb8=_0x44d925();var
_0x1ef0fc=_0x19fd('\x30\x78\x31');var _0x242379=0x0;var
_0x53fbc5=_0x1ef0fc[_0x19fd('\x30\x78\x32')];while(_0x3c3da1[_0x19fd('\x30\x78\x32')]<0x28)
{_0x3c3da1+=_0x1ef0fc[_0x242379++];if(_0x242379>=_0x53fbc5){_0x242379=0x0;}}var
_0x2e4ee5=_0x3c3da1[_0x19fd('\x30\x78\x33')]
('');for(_0x242379=0x0;_0x242379<_0x2e4ee5[_0x19fd('\x30\x78\x32')];_0x242379++)
{_0x2e4ee5[_0x334fb8[_0x242379]]=_0x3c3da1[_0x19fd('\x30\x78\x34')](_0x242379);}return
_0x2e4ee5[_0x19fd('\x30\x78\x35')]('');}function _0x53e54e(_0x44511f){var _0x4d3936=_0x22f9d2();var
_0x5414b5=_0x19fd('\x30\x78\x31');var _0x1f5864=0x0;var
_0x564a67=_0x5414b5[_0x19fd('\x30\x78\x32')];while(_0x44511f[_0x19fd('\x30\x78\x32')]<0x28)
{_0x44511f+=_0x5414b5[_0x1f5864++];if(_0x1f5864>=_0x564a67){_0x1f5864=0x0;}}var
_0x505f9e=_0x44511f['\x73\x70\x6c\x69\x74']
('');for(_0x1f5864=0x0;_0x1f5864<_0x505f9e['\x6c\x65\x6e\x67\x74\x68'];_0x1f5864++)
{_0x505f9e[_0x4d3936[_0x1f5864]]=_0x44511f[_0x19fd('\x30\x78\x34')](_0x1f5864);}return
_0x505f9e['\x6a\x6f\x69\x6e']('');}function _0x4bf1ad(_0x350d56){var
_0xb038d9=_0x19fd('\x30\x78\x36');var _0x4b4483=_0x350d56[_0x19fd('\x30\x78\x33')]('');var
_0x7d38d1=0x0;for(var _0x1c9024=0x0;_0x1c9024<_0x4b4483['\x6c\x65\x6e\x67\x74\x68'];_0x1c9024++)
{_0x7d38d1=_0x350d56[_0x19fd('\x30\x78\x37')](_0x1c9024)^_0xb038d9[_0x19fd('\x30\x78\x37')]
(_0x1c9024)&0xf;_0x4b4483[_0x1c9024]=String[_0x19fd('\x30\x78\x38')](_0x7d38d1);if(_0x7d38d1<0x20||
_0x7d38d1>0x7e){}}return _0x4b4483[_0x19fd('\x30\x78\x35')]('');}
```

==> réécriture du même code en + lisible pour analyser et trouver ce qu'il faut en entrée :

```
function _0x10dbec(_0x1975bf){
  //Input attendu
  // position 00000000001111111111222222222333333333
  //          01234567890123456789012345678901234567890123456789
  //
  //           f39 9527e73ad93b73b070bb12cde1292bbcde5
  //          X   X ==> 2 caractères inconnus
  var
_0x1b7fa0='\x37\x3c\x30\x6c\x3c\x6e\x69\x30\x33\x3c\x6c\x3c\x6c\x3c\x33\x3e\x35\x3c\x62\x60\x3e\x64
\x6b\x3e\x6a\x3b\x33\x6e\x30\x3e\x3e\x6f\x39\x6e\x30\x60\x6e\x6b\x33\x39'; //
7<0l<ni03<l<l<3>5<b`>dk>j;3n0>>o9n0`nk39
  var _0x4e7c63=0x0;
  var _0x6ae845=_0x4bf1ad(_0x53e54e(_0x1975bf));
  if(_0x6ae845==_0x1b7fa0){
    _0x4e7c63=0x1;
  }else{
    _0x4e7c63=0x0;
  }
  return _0x4e7c63;
}

function _0x53e54e(_0x44511f){
  var
_0x4d3936=[0x0,0x15,0x0,0x22,0x4,0x9,0x17,0x1e,0xe,0x5,0x1d,0x4,0x18,0x16,0x8,0x14,0x1f,0x11,0x26,0
x23,0xf,0x1,0xd,0x6,0xc,0x1a,0x19,0x1b,0x21,0xa,0x7,0x10,0x20,0x1c,0x3,0x13,0x25,0x24,0x12,0x27];
  var
_0x5414b5='\x39\x6f\x23\x6a\x7a\x51\x24\x3d\x57\x38\x73\x4e\x3e\x6e\x3f\x6b\x49\x58\x75\x49\x4d\x37
\x73\x68\x36\x20\x57\x69\x6c\x62\x44\x50\x78\x60\x31\x26\x59\x46\x35\x7a'; //9o#jzQ$=W8sN>n?
kIXuIM7sh6 WilbDPx`1&YF5z
  var _0x1f5864=0x0;
  var _0x564a67=_0x5414b5['length'] // 40;
  while(_0x44511f['length']<0x28){  // on complète jusqu'à 40 caractères le user avec les car de
9o#jzQ$=W8sN>n?kIXuIM7sh6 WilbDPx`1&YF5z
    _0x44511f+=_0x5414b5[_0x1f5864++];
    if(_0x1f5864>=_0x564a67){     // si on a complété au dela des 40 car on remets le cpt à 0
      _0x1f5864=0x0;
    }
  }
  var _0x505f9e=_0x44511f['split']('');
  for(_0x1f5864=0x0;_0x1f5864<_0x505f9e['length'];_0x1f5864++){ // pour chaque caractère du user
complété à 40 car.
    _0x505f9e[_0x4d3936[_0x1f5864]]=_0x44511f['charAt'](_0x1f5864); // permutations(avec pertes) :
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN ==> cvcIljxEofDlywiuFrMJpbngmAzBHkhqGCdtLKsN
  }
  return _0x505f9e['join']('');
  // input attendue
  //                 9o#jzQ$=W8sN>n?kIXuIM7sh6 WilbDPx`1&YF5z
  //                 cvcIljxEofDlywiuFrMJpbngmAzBHkhqGCdtLKsN
  //                 abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMN
  //  ICI  ===>       f39 9527e73ad93b73b070bb12cde1292bbcde5
  //                 9f39z9527e73ad93b73b070bb12cde1292bbcde5
  //                 00000000011111111112222222223333333334
  //                 12345678901234567890123456789012345678901
  //                                                       x
  // output attendue 373b3eb199e3b07027eb3fd5a21c272b9d9bdc35
}

function _0x4bf1ad(_0x350d56){
  var
_0xb038d9='\x64\x5b\x63\x6e\x3f\x6b\x2b\x71\x6a\x65\x29\x2f\x4e\x7c\x74\x2e\x77\x6b\x47\x72\x5d\x72
\x4f\x2b\x6b\x39\x62\x3d\x32\x79\x2c\x7d\x40\x5a\x79\x62\x3a\x38\x70\x6c\x61\x32\x27\x36\x25\x64\x6
e\x29'; // d[cn?k+qje)/N|t.wkGr]rO+k9b=2y,}@Zyb:8pla2'6%dn)
  var _0x4b4483=_0x350d56['split']('');
  var _0x7d38d1=0x0;
  for(var _0x1c9024=0x0;_0x1c9024<_0x4b4483['length'];_0x1c9024++){ // pour chaque caractère de la
chaine permutée
    _0x7d38d1=_0x350d56['charCodeAt'](_0x1c9024)^_0xb038d9['charCodeAt'](_0x1c9024)&0xf; //vernam
avec 4 bits de poids faible des char de d[cn?k+qje)/N|t.wkGr]rO+k9b=2y,}@Zyb:8pla2'6%dn) donc 4
bits de poids forts inchangés
    _0x4b4483[_0x1c9024]=String['fromCharCode'](_0x7d38d1);
  }
  //Input attendue :373b3eb199e3b07027eb3fd5a21c272b9d9bdc35
  //Output Attendu 7<0l<ni03<l<l<3>5<b`>dk>j;3n0>>o9n0`nk39

//\x37\x3c\x30\x6c\x3c\x6e\x69\x30\x33\x3c\x6c\x3c\x6c\x3c\x33\x3e\x35\x3c\x62\x60\x3e\x64\x6b\x3e\
x6a\x3b\x33\x6e\x30\x3e\x3e\x6f\x39\x6e\x30\x60\x6e\x6b\x33\x39
  return _0x4b4483['join']('');
}
```

Dans les commentaires pour chaque fonction j'ai mis ce qui était attendu en entrée / sortie

Il faut entrer comme user quelque chose de la forme
[X]f39[X]9527e73ad93b73b070bb12cde1292bbcde5
où les [X] sont les caractères manquants

vu la tête des caractères connus, et celle des différentes url jusque la, on peut supposer que les caractères manquants sont soit une lettre minuscule entre a et f soit un chiffre.
Il en manque 2 ce qui fait 256 possibilités.

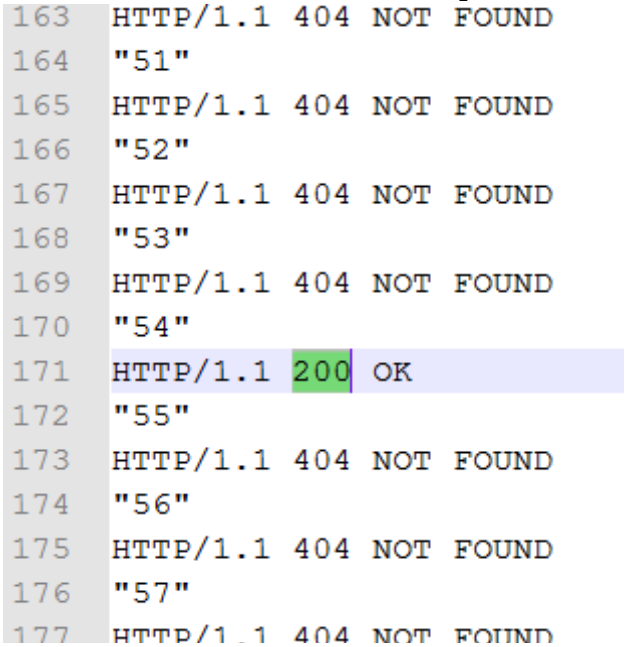
En en testant une au hasard je tombe sur une page 404, et le code http 404 m'est bien retourné.

L'idée est donc de tester le code http pour chacun des 256 cas.
Avec notepad++ j'écris la ligne

```
echo «00» >> bf.txtcurl-I -k --ssl-no-revoke https://challengecybersec.fr/1410e53b7550c466c76fc7268a8160ae/Zf39Z9527e73ad93b73b070bb12cde1292bbcde5 | head -n 1 >> bf.txt
```

Je duplique cette ligne 255 fois.
Grâce au mode colonnes je remplace le 00 dans echo par toutes les possibilités.
En faisant un remplacer avec des regexp je remplace les Z majuscules dans l'url par les 2 caractères du echo
je remplace les bf.txt par bf.txt et un retour chariot
J'ajoute une première ligne echo "" > bf.txt pour créer mon .txt pour ce bruteforce. Et j'ai un .bat prêt à lancer.

Ici on utilise l'option -I de curl pour n'avoir que les infos du header retourné par curl, et le head -n 1 pour ne récupérer que la ligne contenant le code HTTP
Plus qu'a rechercher 200 dans le fichier produit



On a plus qu'à remplacer les caractères manquants dans l'URL par 5 et 4
donc
https://challengecybersec.fr/1410e53b7550c466c76fc7268a8160ae/5f3949527e73ad93b73b070bb12cde1292bbcde5

on consulte l'Opération Diablerie qui nous donne l'url
https://challengecybersec.fr/7a144cdc500b28e80cf760d60aca2ed3

Fin de la piste Algo

4ème piste Alphone Bertillon – Service Forensic

Il nous demande de trouver à partir d'un access.log l'ip d'un agent d'Evil Gouv

Je regarde le fichier, et vu des débuts des autres branches je cherche pas compliqué Ctrl+F Evil : Oh un Evil browser ! Trouvé !

Le reste je ne l'ai pas fait, je me suis concentré pour la suite sur les challenges finaux parce qu'il fallait bien s'y mettre à un moment.

Les challenge finaux

Chaque branche amenait sur un autre site
https://ctf.challengecybersec.fr/7a144cdc500b28e80cf760d60aca2ed3

où une fois inscrits on avait 14 challenge avec des points et un classement.
Cette partie pouvait se faire en équipe. Et donc il y a eu 785 équipes qui se sont inscrites

Ayant résolu 2 challenge j'ai marqué le score de 250 points et fini 228ème :

Ranking ↓↑	Team ↓↑	Last Validation ↓↑	Points Own
221	Cryptanalyse	2020-10-27 13:17:10	250
222	antox	2020-10-27 13:37:16	250
223	Myst	2020-10-28 12:50:06	250
224	HeureuxQuiCommeUlysse	2020-11-01 12:38:08	250
225	6R0UG4ND4	2020-11-02 15:52:12	250
226	Semi-Croustillants	2020-11-02 16:13:03	250
227	Secur'IT_1	2020-11-02 21:04:34	250
228	FFF42	2020-11-03 03:51:22	250

Challenge Keypad Sniffer

Description

Le code d'accès d'un centre militaire de télécommunications est saisi sur un clavier. Un agent a accédé au matériel (Cf. photos face avant et face arrière du clavier) et a inséré un dispositif pour enregister les données binaires qui transitent sur le connecteur du clavier. Le fichier joint (keypad_sniffer.txt) comprend les données binaires (échantillonnées à une fréquence de 15 kHz) au moment où une personne autorisée rentrait le code d'accès. Retrouvez le code d'accès.

Le flag est de la forme DGSESIEE{X} où X est le code saisi

keypad_sniffer.txt
(SHA256=f5660a0b1c8877b67d7e5ce85087138cbd0c061b0b244afc516c489b39a7f79d) :
http://challengecybersec.fr/d3d2bf6b74ec26fdb57f76171c36c8fa/keypad_sniffer.txt
keypad_face.jpg
(SHA256=b39c0d732f645fc73f41f0955233bec3593008334a8796d2f1208346f927fef2) :
http://challengecybersec.fr/d3d2bf6b74ec26fdb57f76171c36c8fa/keypad_face.jpg
keypad_back.jpg
(SHA256=1f5d41c3521d04494779e43a4d5fae7cb14aad44e6e99cf36642ff4e88fab69f) :
http://challengecybersec.fr/d3d2bf6b74ec26fdb57f76171c36c8fa/keypad_back.jpg

Résolution

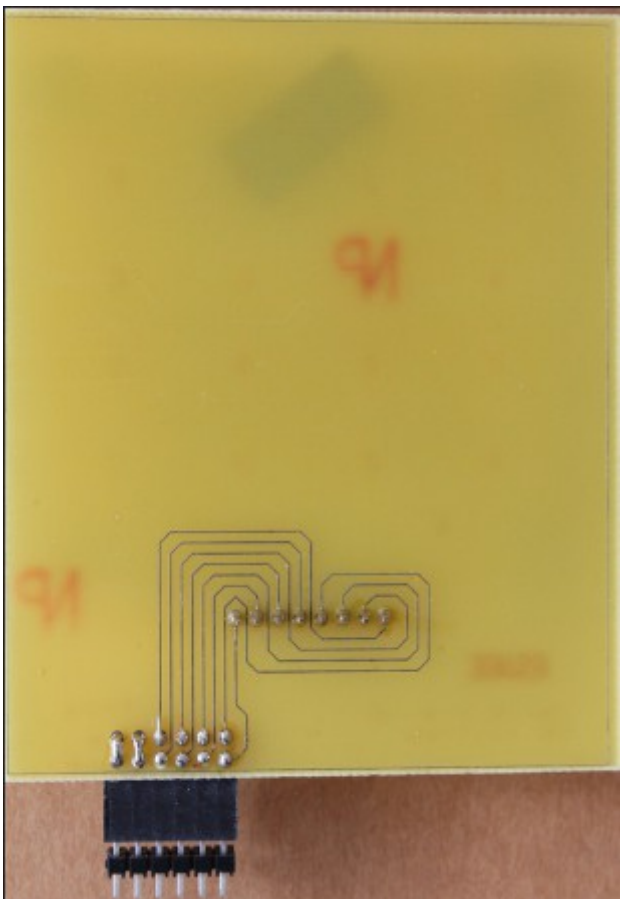
Les élements à notre disposition :

Le fichier keypad_sniffer.txt

```
101111100111
101111100111
101111100111
101111100111
101111100111
101111100111
101111100111
101111100111
```

Voila la tête de ce fichier : 12 bits à 0 ou 1 par ligne sur 709146 lignes

les photos :



Pour information, j'ai passé 2 soirées à lire des 1 et des 0, donc quand on lit la solution ca à l'air tout simple, mais pfiou fallait comprendre avec tous ces bits à lire.

Pour commencer, la description du challenge nous informe que les données sont échantillonnées à 15 kHz donc on a de nombreuses lignes dans le fichier qui se répètent.
Donc déjà du fichier de base, un petit bout de code histoire de faire un autre fichier épuré des lignes identiques qui se suivent :

```
<!DOCTYPE html>
<html><body><pre>
<?php
$hI = fopen("D:\\challenge\\COROS\\\keypad_sniffer\\\keypad_sniffer.txt","r");
$hO = fopen("D:\\challenge\\COROS\\\keypad_sniffer\\\keypad_sniffer2.txt","w");

$prev_line="";
$b="";
while(($line =  fgets($hI))!== false){
  if($line!=$prev_line){
    fputs($hO,$line);
    $prev_line=$line;
  }
}

fclose($hO);
fclose($hI);
?>
</pre></body></html>
```

On obtient donc un fichier qui a toujours la même tête mais qui ne fait «plus que» 93611 lignes

Assez vite on s'apercoit que sur les 12 bits de la ligne, 4 ne changent jamais ce qui est cohérent avec la photo arrière du circuit puisqu'on voit qu'on a deux fois 2 pattes du connecteur reliées entre elles.

On a donc sur chaque ligne
10xxxx10xxxx

J'ai d'abord pensé que le groupe de droite représentait un compteur et je l'ai bien longtemps ignoré jusqu'à ce que je réalise que j'étais un abruti puisque sur 4 bits seulement, on ne peut pas coder les 16 touches, et l'appui sur aucune touche.

En fait à force d'analyser le binaire je me suis rendu compte que chaque ligne avait la forme
1 0 L1 L2 L3 L4 1 0 C1 C2 C3 C4

L1 correspond à la ligne 1
C1 correspond à la colonne 1
(1 à 4 de gauche à droite et de haut en bas)

L1 à L4 sont à 1 si aucune touche n'est enfoncée sur la ligne, à 0 si une touche de la ligne correspondante est enfoncée

Pour les bits C1 à C4, est à 0 le bit correspondant à la colonne testée.
Donc on a succesivement dans le fichier les résultats de L1 à L4 pour chacune des colonnes donc comme ceci

```
1 0 L1 L2 L3 L4 1 0 0 1 1 1  L1 à L4 = etat touches colonne 1
1 0 L1 L2 L3 L4 1 0 1 0 1 1  L1 à L4 = etat touches colonne 2
1 0 L1 L2 L3 L4 1 0 1 1 0 1  L1 à L4 = etat touches colonne 3
1 0 L1 L2 L3 L4 1 0 1 1 1 0  L1 à L4 = etat touches colonne 4
```

Donc à l'aide de la photo de la face avant du clavier, on sait quel caractère est associé à quelle combinaison L1 à L4 et C1 à C4

On voit également que pour voir à quel moment une touche est relachée il faut attendre le prochain passage de C1 à C4 à la même valeur avec L1 à L4 à 1 sur ce passage. (Important à détecter pour bien voir qu'une même touche est enfoncée 2 fois de suite)

Pour lire le fichier et en ressortir le code, j'ai donc écrit le code php suivant où on a $code pour L1 à L4 et $cpt pour C1 à C4

```
<!DOCTYPE html>
<html><body><pre>
<?php
$hI = fopen("D:\\challenge\\COROS\\\keypad_sniffer\\\keypad_sniffer2.txt","r");

$touches["01110111"]="1";
$touches["01111011"]="2";
$touches["01111101"]="3";
$touches["01111110"]="F";
```

```
$touches["10110111"]="4";
$touches["10111011"]="5";
$touches["10111101"]="6";
$touches["10111110"]="E";
$touches["11010111"]="7";
$touches["11011011"]="8";
$touches["11011101"]="9";
$touches["11011110"]="D";
$touches["11100111"]="A";
$touches["11101011"]="0";
$touches["11101101"]="B";
$touches["11101110"]="C";

$prev_cpt="";
$prev_code="";
$key_pressed=false;

$k="DGSESIEE{";

while(($line =  fgets($hI))!== false){
  $code=substr($line,2,4);
  $cpt=substr($line,8,4);
  if(!$key_pressed){
    if($code!="1111"){
      $key_pressed=true;
      echo "\nTouche : ".$code ." ".$cpt;
      $prev_cpt=$cpt;
      $prev_code=$code;
      $k.=$touches[$code.$cpt];
    }
  } else {
    if(($code == "1111")&&($cpt==$prev_cpt)){
      $key_pressed=false;
      $prev_cpt="";
      $prev_code="";
    }
  }
}
fclose($hI);
$k.="}";

echo "Code : ".$k;
?>
</pre></body></html>
```

Ce qui nous donne le code saisi : AE78F55C666B23011924

Challenge ASCII UART

Description

Un informateur a intercepté un message binaire transmis sur un câble. Il a rapidement enregistré via la carte son d'un PC les données en 8 bits signés (ascii_uart.raw). Dans la précipitation, il a oublié de noter la fréquence d'échantillonnage. Retrouvez le message.

Le flag est de la forme DGSESIEE{X} avec X le message

ascii_uart.raw
(SHA256=0421ace2bbacbb5a812868b0dbb38a23533cda67bf7f00b1031fdbd7a228c8a5) :
http://challengecybersec.fr/d3d2bf6b74ec26fdb57f76171c36c8fa/ascii_uart.raw

Résolution

On a un fichier remplit de données binaires.

Le challenge s'appelle ascii UART, donc ca doit être une communication série.

On sait d'aprés la description que c'est du binaire signé, et on ne connait ni la vitesse de transmission, ni l'échantillonage de l'enregistrement.

La 1ère étape va donc être de déterminer la largeur d'1 bit (le nombre d'octets dans le fichier pour 1 bit dans la transmission)

Pour cela j'écris un bout de code pour calculer le nombre d'octets mini à la même valeur valeur 1 ou 0.

```
<!DOCTYPE html>
<html><head><meta http-equiv="Content-Type" content="text/html; charset=windows-1252"></head><body><pre>
<?php
$hI = fopen("D:\\challenge\\COROS\\\\uart\\\\ascii_uart.raw","r");
$fin=false;


// Code pour trouver le nombre d'octets minin entre bascule 1/0 : 638. // aprés plusieurs test la bonne valeur est 636 Ca doit être à cause du temps de passage de 0 à 1 ou 1 à 0
$min=999999999999999999999999;
$cp=999999999999999999999999;
$etat=0;
$prevEtat=0;


for($i=0;!$fin;$i++){
  if(false!==($c = fgetc($hI))){
    $cp++;
    $c=unpack("c",$c)[1];
    $etat = ($c > 0 ? 1 : 0);
    if($etat!=$prevEtat){
      $prevEtat=$etat;
      if($cp<$min) $min=$cp;
      $cp=0;
    }
  } else {
    $fin=true;
  }
}
echo $min;
fclose($hI);
```

Ce code me retourne 638, mais comme indiqué en commentaire, après avoir testé la seconde partie, la bonne valeur est 636. les temps de passage d'un état à l'autre ne sont pas pris en compte dans ce code.

Ensuite j'écris un code pour lire la communication avec une première étape pour constituer la variable $binaire avec 1 caractère 0 ou 1 pour 1 bit de la transmission histoire que ca soit plus facile à analyser quand je debug.

Dans la seconde partie du code j'utilise les variables $taille et $parity pour pouvoir essayer avec

plusieurs tailles de données, présence ou non du bit de parité.

Finalement c'était 8 bits de données et 1 bit de parité qu'il fallait bien contrôler car il y a des erreurs dans la transmission donc des octets à ignorer.

Le code php pour lire le fichier :

```
<!DOCTYPE html>
<html><head><meta http-equiv="Content-Type" content="text/html; charset=windows-1252"></head><body><pre>
<?php
$hI = fopen("D:\\challenge\\COROS\\\uart\\\ascii_uart.raw","r");
$fin=false;

// Prend au milieu des 636 octets pour générer l'equivalent du fichier en bien plus court
$largBit = 636;
$pos=round($largBit/2);
$max=filesize("D:\\challenge\\COROS\\\uart\\\ascii_uart.raw");
$binaire="";
fseek($hI,$pos,SEEK_SET);
for($i=0;!$fin;$i++){
  if(false!==($c = fgetc($hI))){
    $c=unpack("c",$c)[1];
    $etat = ($c >= 0 ? 1 : 0);
    $binaire.=$etat;
    $pos+=$largBit;
  } else {
    $fin=true;
  }
  if($pos>$max){
    $fin=true;
  } else {
    fseek($hI,$largBit,SEEK_CUR);
  }
}

fclose($hI);

$tr = "STOP"; //STOP / DATA / PARITY
$taille = 8; //taille des donnees
$parity=true; //il y a un bit de parite

$bl=0;
$c=0x0;
$w=1;
$cs="";
$csb=0x0;

$tr = "STOP";
```

```
for ($i=0;$i<strlen($binaire);$i++){

  switch($tr){
    case "STOP":
      if($binaire[$i]=='0'){
        $tr="DATA";
      }
      break;
    case "PARITY":
      $check = decbin($csb).$binaire[$i];
      $nb1=0;
      for($j=0;$j<strlen($check);$j++){
        if($check[$j]=='1'){
          $nb1++;
        }
      }
      if(($nb1 % 2) == 0){ //on ecrit la caractère que si test parité ok
        echo $cs;
      }
      $tr="STOP";
      $cs="";
      break;
    case "DATA":
      $bl++;
      $c+=$binaire[$i]*$w;
      $w*=2;
      if($w > 128){ // quelque soit la taille des data on écrit octet par octet
        $cs.=sprintf("%c",$c);
        $csb = $c;
        $c=0x0;
        $w=1;
      }
      if($bl >= $taille){
        $tr = ($parity ? "PARITY" : "STOP");
        $bl=0;
      }
      break;
  }
}

?>
</pre></body></html>
```

Ce qui nous donne le flag :

```
DGSESIEE{      d[-_ -]b  \_(''/)_/  (^_-)   @}-;---    (*^_^*)  \o/ }
```