EXPLOATAREA VULNERABILITĂȚILOR WEB - SQL INJECTION ATAC

Autor: Eugeniu DAVID, student SI-141

Universitatea Tehnică a Moldovei, catedra Automatică și Tehnologii Informaționale

ABSTRACT: *Injectarea SQL este un atac asupra aplicațiilor WEB, care constă în „injectarea" unei interogări SQL printr-un imput de date de la client către aplicație. Acest tip de atac rămâne până în prezent unul din cele mai des întâlnite și este una din cele mai exploatate vulnerabilități ale aplicațiilor WEB. Un atac petrecut cu succes poate permite atacatorului să citească informații sensibile din baza de date, să modifice baza de date, să execute operații de administrare asupra bazei de date și în unele cazuri poate permite executarea comenzilor asupra sistemului de operare care găzduiește baza de date.*

În această lucrare sunt descrise tipurile acestor atacuri și este inițiat un astfel de atac utilizând un sistem automat de detectare a aplicațiilor vulnerabile Google Dorks.

CUVINTE CHEIE*: aplicații web, baze de date, vulnerabilități, SQL injection, SQL Map, Google Dork.*

Introducere

La momentul actual aplicațiile web rulează, în spate, o bază de date care stochează toată informația, aceasta permițând generarea paginilor dinamic. Aplicația primește de la utilizator niște date după care generează o cerere la baza de date. Pentru generarea cererilor către baza de date cel mai des se utilizează limbajul SQL (Structured Query Language).

Injectarea SQL este o vulnerabilitate ce apare atunci când datele primite de la utilizatori nu sunt prelucrare corect sau nu sunt prelucrate deloc. Ca consecință – un răufăcător poate schimba cererea către baza de date, astfel încât să extragă din baza de date informație confidențială.

Acest tip de atac poate fi săvârșit asupra aplicațiilor ce lucrează cu orice tip de SGBD (MySQL, MS SQL, Oracle), logica de atac va fi aceeași doar sintaxa interogărilor va varia.

1. Ce este o bază de date?

O bază de date (BD), reprezintă o modalitate de stocare a unor informații și date pe un suport extern (un dispozitiv de stocare), cu posibilitatea extinderii ușoare și a regăsirii rapide a acestora.

Cel mai răspândit tip de baze de date este cel relațional, în care datele sunt memorate în tabele. Pe lângă tabele, o bază de date relațională mai poate conține: indecși, proceduri stocate, declanșatori, utilizatori și grupuri de utilizatori, tipuri de date, mecanisme de securitate și de gestiune a tranzacțiilor etc.

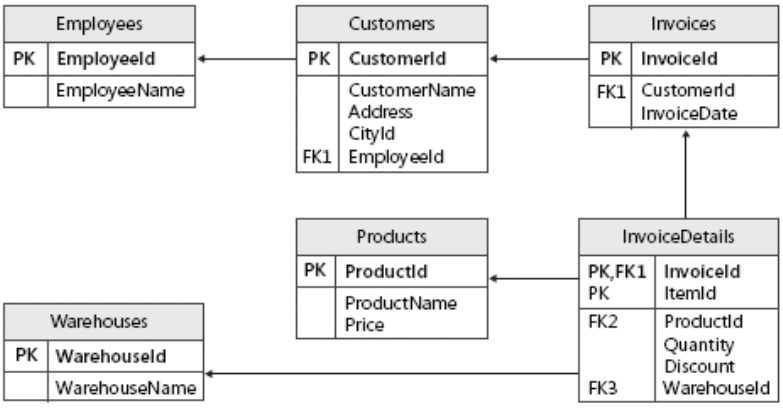

Figura 1 – Structura unei baze de date

Cea mai simplă interogare la o bază de date este: SELECT * FROM Costumers.

Această instrucțiune ar afișa tot conținutul tabelului Costumers. Astfel, atunci când un utilizator, accesează o pagină sau cere afișarea unui oarecare conținut de pe aplicația-web, se generează o cerere către baza de date ce asigură această funcție.

2. Scurtă descriere a atacului

Atacul *Injectarea SQL* constă în generarea interogărilor direct către baza de date, astfel, pas cu pas, determinând tipul de bază de date din dosul aplicației-web, denumirea schemelor, coloanele vulnerabile, denumirea tabelelor și ulterior extragerea datelor din baza de date.

Pentru a determina dacă o aplicație este vulnerabilă la atac trebuie să ne asigurăm că aceasta are în spate o bază de date, să determinăm paginile care accesează baza de date pentru a introduce sau a extrage informații.

Cele mai des utilizate tehnologii pentru realizarea legăturii unei baze de date cu aplicația-web sunt PHP sau ASP. Deci, pentru a determina dacă aplicația are în spate o bază de date și este vulnerabilă, se accesează pe rând toate paginile și se urmărește dacă în adresa paginii se conține următoare secvență:

asp?id= sau php?id=

Pagina ce conține o sintaxă asemănătoare, face legătura dintre aplicație și baza de date, prin POST sau GET. Pentru determinarea paginilor ce accesează baza de date, în cazul când aplicația este vulnerabilă, la finalul adresei se introduce un simbol intrus, ca exemplu, un apostrof.

http://site.com/pagina-vulnerabila.asp?id=1'

Dacă în urma executării obținem un astfel de răspuns:

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 ''' at line 1

Mesajul prezintă primul indice că aplicația este vulnerabilă la injectare. În continuare este necesar de determinat numărul de coloane din baza de date. Aceasta se face utilizând instrucțiunea ORDER BY sau GROUP BY:

http://site.com/pagina-vulnerabila.asp?id=1 ORDER BY 1

Instrucțiunea se execută incrementând cifra 1, până se obține o eroare care ne spune că coloana este necunoscută. Numărul de coloane din baza de date va fi egal cu numărul din interogarea cu eroare minus 1.

Dacă se cunoaște numărul de coloane, se determină coloanele vulnerabile la injectare. Pentru aceasta se face un UNION SELECT la toate coloanele din baza de date. Inițial se introduce un NULL (-) în fața interogării, în unele cazuri poate fi necesară introducerea unui dublu NULL la finele interogării (--), aceasta depinde de tipul SGBD-ului și versiunea folosită.

http://site.com/pagina-vulnerabila.asp?id=-1 UNION SELECT 1,2,3,...,n--

După rularea acestei secvențe pe pagină vor fi afișate niște cifre, acestea indică numărul coloanei/coloanelor vulnerabile. Pentru extragerea informației, în locul numărului coloanei vulnerabile se va introduce interogări către baza de date.

De asemenea, este necesar de a determina versiunea tehnologiei utilizate. La diferite versiuni, poate varia sintaxa de interogare. Deci, dacă a fost găsită coloana n vulnerabilă, pentru determinarea versiunii tehnologiei se modifică sintaxa cererii:

http://site.com/pagina-vulnerabila.asp?id=-1 UNION SELECT 1,2,3,...,version()--

Versiunea determinată se analizează pentru a cunoaște specificul sintaxei. Ulterior, în loc de version() se introduce database(), pentru a determina denumirea bazei de date:

http://site.com/pagina-vulnerabila.asp?id=-1 UNION SELECT 1,2,3,...,database()--

și numele tabelelor:

http://site.com/pagina-vulnerabila.asp?id=-1 UNION SELECT 1,2,3,...,group_concat(table_name) from information_schema.tables where table_schema=database()--

Astfel, prin instrucțiuni similare, se determină denumirea coloanei, în special coloanele ce conțin login și parole.

3. SQL Injection în practică

Pentru aplicarea în practică se va folosi aplicația SQL Map, aceasta fiind una din cele mai puternice aplicații care permite verificarea aplicațiilor-web la SQL injectare.

La fel, se va folosi un Google Dork pentru găsirea unei aplicații vulnerabile. Un Google Dork reprezintă un fel de amprentă, o secvență de cod care permite depistarea diferitor aplicații vulnerabile la diferite tipuri de atacuri. Pe site-ul http://www.allhackingtools.com/ pot fi găsite un număr mare de Google Dorks.

Aplicația SQL Map vine preinstalată pe Kali Linux, un soft specializat pentru teste de penetrare. Aceasta este scrisă în limbajul Python și poate fi rulată pe orice dispozitiv care poate interpreta codul în acest limbaj. Aplicația poate fi descărcată de pe site-ul oficial http://sqlmap.org/ și rulată prin intermediul liniei de comandă (figura nr.2).

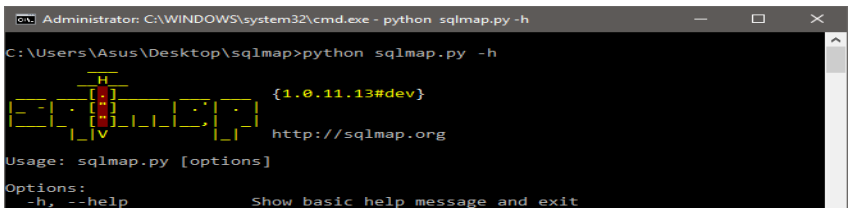

Figura 2 – Rularea sqlmap

Programul dat automatizează tot procesul, de la determinarea dacă aplicația este vulnerabilă, până la extragerea datelor din baza de date și chiar decriptarea acestora. Pentru exemplul dat, o sa rulăm o simplă comandă care va determina denumirea bazei de date a aplicației:

python sqlmap.py -g accinfo.php?cartId= --dbs

Acest proces durează ceva timp și poate fi necesar adăugarea unor noi opțiuni de atac pentru a mări riscul sau nivelul testelor care să fie rulate sau pentru a determina proxy sau random-agent pentru a nu fi detectat.

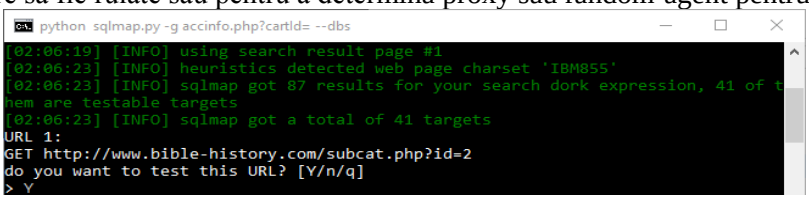

Figura 3 – Detectarea aplicației vulnerabile

La rularea codului de mai sus, putem observa că aplicația oferă un număr vast de informație pentru atacator. Se anunță numărul de aplicații injectabile și permisiunea de a ataca una din acestea. După cum putem observa, utilizând Google Dork-ul, au fost depistate 41 de ținte potențiale. Prima di listă a fost aleasă drept țintă. De la utilizator se cere permisiunea de a începe testul.

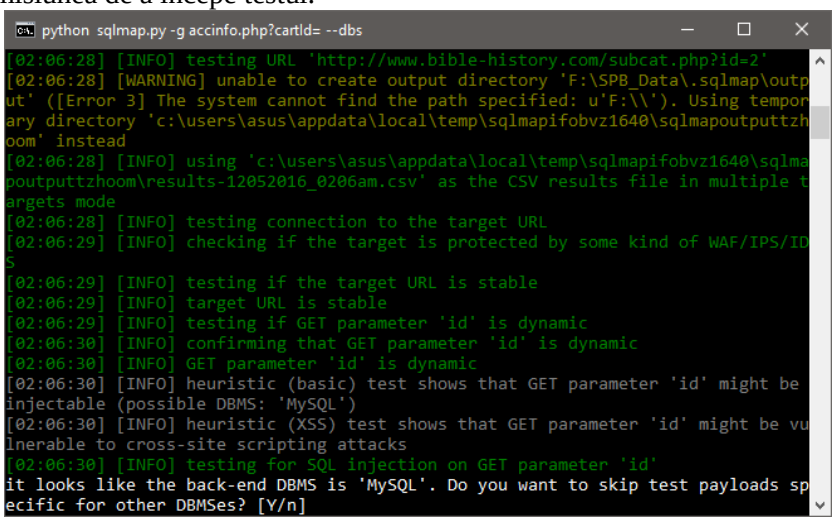

Figura 4 – Depistarea tipului de baza de date

După cum se observă, aplicația în timp ce rulează testele, afișează informația despre testul ce se rulează și unele informații care ar putea fi utile pentru utilizator. În cazul dat, se vede că aplicația este vulnerabilă și la atac corss-site scripting.

După depistarea tipului de bază de date se întreabă utilizatorul dacă acesta dorește să ruleze și celelalte teste posibile. La următoare etapă se va determina versiunea bazei de date utilizate și parametrul injectabil (figura nr. 5).

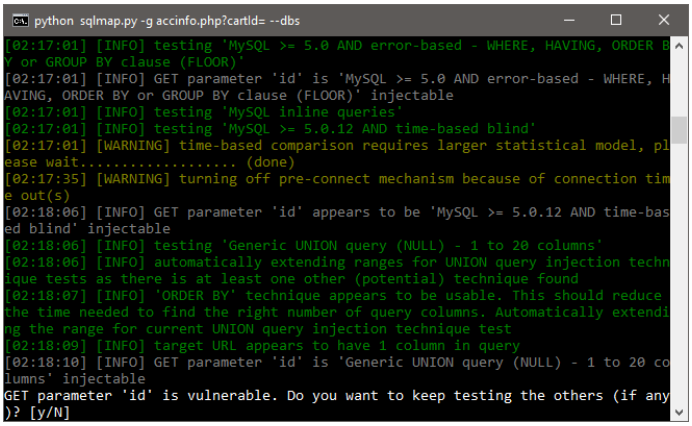

Figura 5 – Determinarea versiunii și parametrului injectabil

De asemenea, aplicația afișează informația despre atacul săvârșit, care a fost parametrul injectat, tipul atacului, titlul și payload-ul utilizat, rezultatul atacului, denumirile bazelor de date prezente în aplicația dată (figura nr.6).

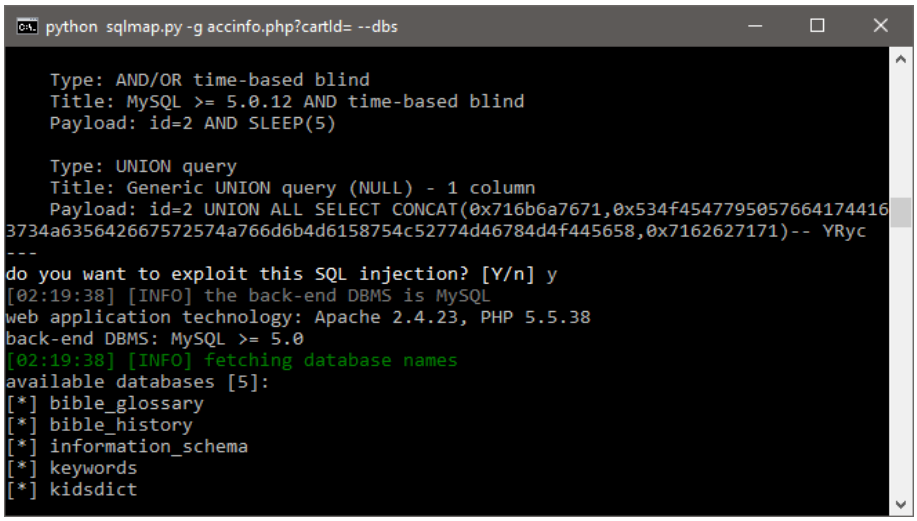

Figura 6 – Rezultatul atacului

Ca rezultat poate fi solicitată afișarea unei baze de date sau poate fi determinat tabelul și coloana care conține informația despre utilizatori sau despre administrator etc.

Concluzii

Se poate observa cât este de ușor de a face un astfel de atac. Utilizând aplicațiile specializate în astfel de atacuri, nici nu este necesar cunoașterea limbajului SQL. Este suficient un simplu tutorial care poate fi găsit chiar pe site-ul dezvoltatorilor acestei aplicații și în decursul al câteva minute, oricine poate extrage date critice dintr-o aplicație vulnerabilă. La momentul actual, cu toate sunt utilizate diferite modalități de protecție, încă rămân lacune privind aceste atacuri.

Bibliografie

1. OWASP SQL Injection. https://www.owasp.org/index.php/SQL_Injection
2. sqlmap SQL Injection Demo. http://sqlmap.org/
3. SQL Injection Cheat Sheet. https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
4. Google Dorks. http://www.allhackingtools.com/2015/03/5000-fresh-google-dorks-sql-injection.html