Politechnika Białostocka Bezpieczeństwo Aplikacji Internetowych	Prowadzący: dr inż. Maciej Brzozowski
Temat: PS-5 SQL Injection 1. Dominik Adrian Kruk 2. Mateusz Matocha	Ocena:

1. W jaki sposób sprawdzić podatność na SQL Injection?

Należy znaleźć stronę na której możemy wprowadzać dane do formularzy. W formularzu zamiast danych podajemy w polach login i hasło „login or 1=1". Strona podatna na ataki tego typu, umożliwi zalogowanie, gdyż po stronie serwera do bazy zostanie wysłane na przykład takie zapytanie: 'Select * from user where login='**login**' **or 1=1--'**. Podobnie zapytanie będzie wyglądało z hasłem, i zwrócą użytkownika o podanym loginie i haśle.

2. W jaki sposób przy wykorzystaniu SQL Injection sprawdzić możliwość wykonywania zapytań zagnieżdżonych?

W formularzu jak wyżej, należy przekazać jako parametr zapytanie które zostanie wykonane, np. Select z popularnymi nazwami tabel: Users,Customers itp.
np. w parametrze login=**dane','Select * from users,'dane2'** zwróci nam całą listę użytkowników z bazy.

3. W jaki sposób przy wykorzystaniu SQL Injection określić liczbę kolumn w zapytaniu?

Poprzez dodanie klauzuli **union** poprzez wykorzystanie błędu konwersji typów:

Login=**'union slect sum(username) from users—'**
na podstawie zwracanych komunikatów o błędach będziemy w stanie określić błędną ilość kolumn oraz typu podanego do **sum**.

4. Czym charakteryzuje się atak SQLInjection typu blind?

Polega na przesłaniu w formularzu zapytań SQL które zwracają zawsze true oraz false. Gdy pomiędzy stronami dla tych dwóch zapytań będzie różnica, pozwoli to na dalszą analizę na atak typu SQL Injection. Przykładowe zapytania:
login='**login**' **or 1=1--'**
login='**login**' **or 1=2--'**

5. Jak zabezpieczyć się przed SQL Injection – w czym może być pomocne rzutowanie i escapowanie? Czy wyrażenia regularne są dobrym rozwiązaniem?

Escapowanie odbywa się poprzez funkcje dla baz mysql **mysql_escape_string**, Dodaje ona znaki unikowe, zabezpieczając przed SQL Injection. Przed znakiem **‘** wstawiany jest znak **/** uniemożliwiający domknięcie SQL. Np. dla Ciągu znaków ,,Ala ma k’ota” zostanie przekazane ,,Ala ma k/’ota”.

Rzutowanie typów zabezpieczy bazę dla kolumn, które mają inny typ niż String, gdyż właśnie taki typ jest przekazywany do zapytania. Natomiast dla typu integer przekazanie w danych wieku **age=30’ or 1=1;** zostanie zrzutowane tylko 30.

Wyrażenia regularne umożliwiają zabezpieczenie przed SQL Injection poprzez sprawdzenie czy pole zostało wypełnione spodziewanymi danymi np. czy do pola z kodem pocztowym został przekazany kod pocztowy.

- Zabezpieczanie przed atakami może się odbyć poprzez:
  walidację danych – metody walidacji opisane wyżej;
- Ograniczenie długości wprowadzonych danych – sprawdzanie po stronie przeglądarki ale także serwera;
- Parametryzowane wywołanie zapytań – wykorzystanie technologii i frameworków uniemożliwiających bezpośredniego wykonywania zapytań na bazie danych;
- Nadawanie odpowiednich uprawnień dla użytkownika np. jeśli użytkownik nie potrzebuje operacji delete to ta operacja powinna być wyłączona z poziomu bazy danych;
- Nie prezentowanie błędów SQL na stronie internetowej;

6. W jaki sposób można wykorzystać Union oraz opóźnione zapytania?

Klauzula Union została opisana w pkt.3. Umożliwia na podstawie zwracanych błędów określać strukturę bazy danych.
Opóźnione zapytania, podobnie jak w przypadku pkt.4. pomagają stwierdzić czy przekazane SQL Injection, spowodowało jakieś zmiany. Obserwowanie czasu odpowiedzi serwera dla różnych danych, pozwala na analizę struktury danych i przygotowania spersonalizowanych ataków.

7. Czym jest atak wielofazowy SQL Injection?

Atak wielofazowy SQL Injection polega na przekazaniu do zapisania do bazy danych, np. do pola z Nazwiskiem części zapytania, które można użyć poprzez wyciągnięcie tej wartości i wstawienie do spreparowanego z kilku składowych zapytania. Część zapytania zostanie zapisana w bazie, nie będzie to przechwycone np. przez wyrażenie regularne. Następnie zostaje przekazane poprzez wyciągnięcie tej wartości i wstawienie do zapytania, gdzie wartość pobrana z bazy może nie być walidowana.

8. Jak oceniasz wykorzystanie przy parametryzowaniu zapytań białych i czarnych list? Co jest lepszym rozwiązaniem?

Białe listy przechowują dane o zaufanych domenach/adresach IP. Domeny te posiadają zwykle większe uprawnienia: np. emaile z takich skrzynek nigdy nie trafiają do folderu SPAM. Czarne Listy zawierają domeny, które nie mogę wykonywać żadnych czynności na stronie. Do skutecznej ochrony przed SQL Injection najlepiej jest wykorzystać obie listy: pierwszą by umożliwiać zaufanym domenom na swobodne poruszanie się po stronach, oraz czarnych, by zablokować potencjalne zagrożenia przed wykonywaniem zapytań, lub zwiększyć kontrolę danych.

9. Jak oceniasz wykorzystanie procedur składowanych?

Zastosowanie procedur składowanych wymusza na użytkowniku podającym dane, do przekazania maksymalnej ilości znaków określonego typu, wypełnienia wszystkich pul formularza, gdyż tylko wtedy procedura zostanie wykonana. Ponadto procedura określa typ wyniku wykonania zapytania, co skutecznie uniemożliwi wyciągnięcie niepożądanych danych.

Przykładowo jeśli procedura przyjmuje parametry typu string **login varchar(20)** i hasło **password varchar(20)** a zwracanym typem jest id typu Integer to niemożliwe jest pobranie z select hasła.

10. Jaki wpływ na bezpieczeństwo ma projekt bazy i uprawnienia użytkowników.

W projekcie bazy używanie popularnych nazw tabel takich jak: **User, Customer, Użytkownik, Rezerwacje** itp. Zwiększają szansę na podatność na atak typu SQL Injection. Są to popularne nazwy i takie są przekazywane w raz z atakiem.
Uprawnienia użytkowników powinny być dostosowane do faktycznych potrzeb. Jeżeli użytkownik nie potrzebuje wykonywać operacji delete to ta operacja powinna zostać wyłączona na poziomie bazy.