

Bezpieczeństwo aplikacji Web

Damian Rusinek



A word cloud centered on a dark purple rectangular background, which is itself set against a larger gold background with dark blue diagonal stripes in the top-left and bottom-right corners. The word cloud contains various cybersecurity-related terms in different colors and sizes. The most prominent words are 'attack' in large yellow font, 'vulnerability' in large red font, and 'owasp' in large blue font. Other visible words include 'security', 'misconfiguration', 'sqli', 'crypto', 'csrf', 'php', 'injections', 'authentication', 'exploit', 'websec', 'data', 'protection', 'xss', 'nosqli', 'api', 'net', 'asvs', 'python', 'access', and 'java'.

access
python
asvs security
misconfiguration sqli
csrf attack crypto
vulnerability
php owasp.net API
injections authentication
java websec exploit data protection xss
nosqli

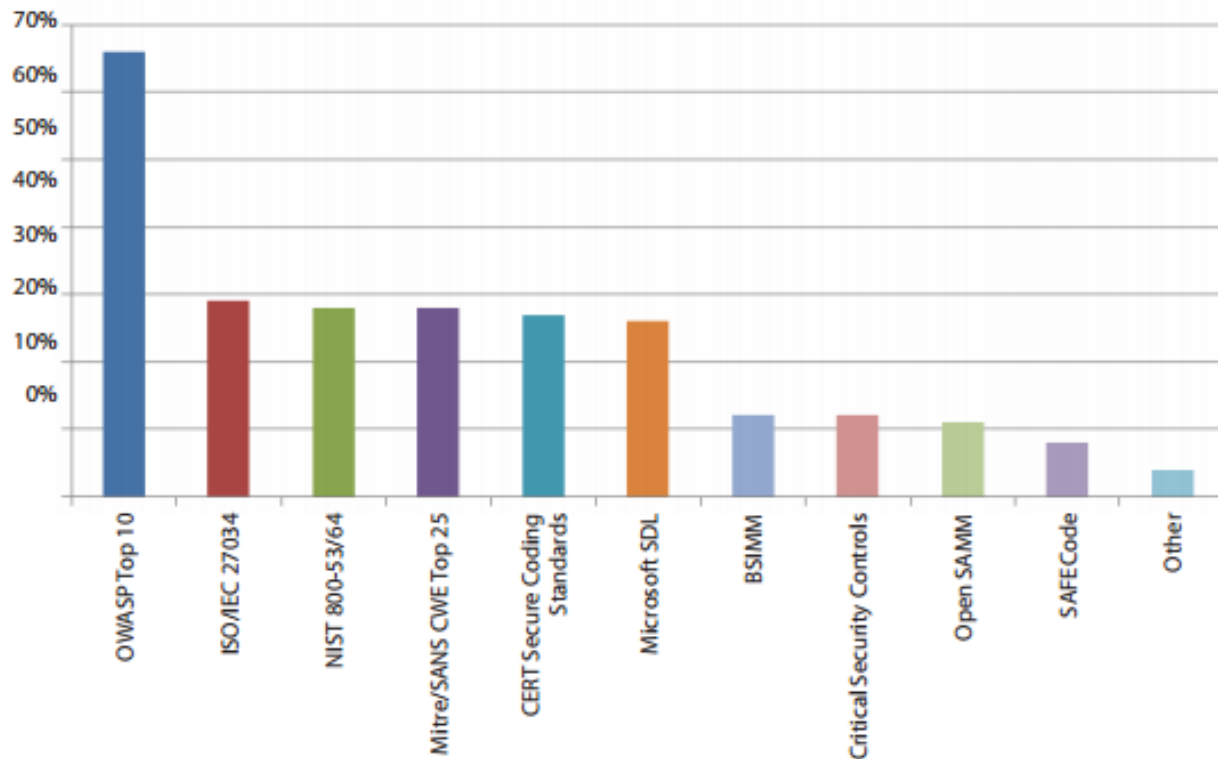
Szkolenie

- Dla **programistów** - nie pentesterów.
- Podejście zorientowane na funkcjonalności, a nie podatności.
- Funkcjonalności i ich słabości pokazane na **przykładach**.
- Dużo **dobrych praktych** i trochę pomocnych **narzędzi**.

Standardy

What application security standards or models do you follow?

Select all that apply.



<https://www.sans.org/reading-room/whitepapers/analyst/2015-state-application-security-closing-gap-35942>

OWASP

- Open Web Application Security Project
- Bardzo dużo podprojektów:
 - Top Ten
 - ASVS
 - Broken Web Application
 - Web Goat
 - Security Knowledge Framework - do niego wrócimy na koniec!

OWASP - Top Ten

- Lista 10 najpopularniejszych krytycznych podatności.
- W 2017 wychodzi nowa wersja, ale skupimy się na wersji poprzedniej (2013).



#	Description	1	2	3	Since
2.1	Verify all pages and resources by default require authentication except those specifically intended to be public (Principle of complete mediation).	✓	✓	✓	1.0
2.2	Verify that all password fields do not echo the user's password when it is entered.	✓	✓	✓	1.0
2.4	Verify all authentication controls are enforced on the server side.	✓	✓	✓	1.0
2.6	Verify all authentication controls fail securely to ensure attackers cannot log in.	✓	✓	✓	1.0
2.7	Verify password entry fields allow, or encourage, the use of passphrases, and do not prevent long passphrases/highly complex passwords being entered.	✓	✓	✓	3.0
2.8	Verify all account identity authentication functions (such as update profile, forgot password, disabled / lost token, help desk or IVR) that might regain access to the account are at least as resistant to attack as the primary authentication mechanism.	✓	✓	✓	2.0
2.9	Verify that the changing password functionality includes the old password, the new password, and a password confirmation.	✓	✓	✓	1.0
2.12	Verify that all suspicious authentication decisions are logged. This should include requests with relevant metadata needed for security investigations.		✓	✓	2.0

OWASP - WebGoat

- Projekt OWASP.
- Aplikacja Web zawierająca wiele podatności bezpieczeństwa w różnych funkcjonalnościach.
- <https://github.com/WebGoat/WebGoat/>
- Najprościej skorzystać z Dockera.

1. Run using Docker

From time to time we publish a new development preview of WebGoat 8 on Docker HUB, you can download this version <https://hub.docker.com/r/webgoat/webgoat-8.0/>. First install Docker, then open a command shell/window and type:

```
docker pull webgoat/webgoat-8.0
docker run -p 8080:8080 webgoat/webgoat-8.0
```


OWASP - BWA

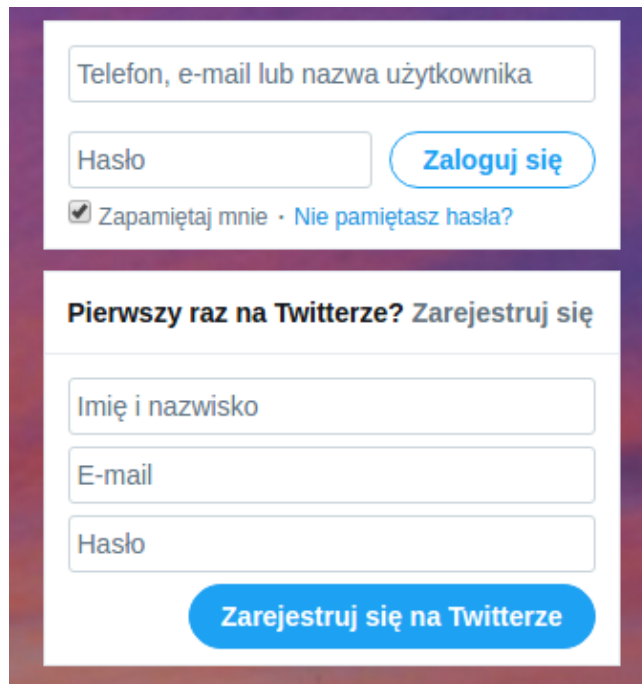
- Projekt OWASP.
- Broken Web Application to obraz VMWare zawierający zestaw aplikacji Web podatnych na znane ataki.
- <https://sourceforge.net/projects/owaspbwa/files/1.2/>

Oprócz powyższych projektów istnieje wiele serwisów, w których można znaleźć maszyny wirtualne z aplikacjami zawierającymi podatności do testowania.

- <https://www.vulnhub.com/>
- <https://www.hackerone.com/>
- <https://www.bugcrowd.com/>
- <https://ctftime.org>

Kiedy pojawia się ryzyko?

- Dane pochodzące od użytkownika.
- Tzw. punkty wejścia do aplikacji.
- Jak stworzyć bezpieczną stronę/serwis WWW?
- Statyczny HTML!



The image shows a screenshot of a web form, likely a login or registration page from Twitter. The form is divided into two main sections. The top section is for logging in, featuring a text input field labeled 'Telefon, e-mail lub nazwa użytkownika', a password input field labeled 'Hasło', and a blue button labeled 'Zaloguj się'. Below the password field is a checkbox labeled 'Zapamiętaj mnie' and a link 'Nie pamiętasz hasła?'. The bottom section is for new users, titled 'Pierwszy raz na Twitterze? Zarejestruj się'. It contains three stacked text input fields labeled 'Imię i nazwisko', 'E-mail', and 'Hasło', followed by a large blue button labeled 'Zarejestruj się na Twitterze'.

Telefon, e-mail lub nazwa użytkownika

Hasło

☒ Zapamiętaj mnie · [Nie pamiętasz hasła?](#)

Pierwszy raz na Twitterze? Zarejestruj się

Imię i nazwisko

E-mail

Hasło

Zarejestruj się na Twitterze

HTTP

- RFC 2616 oraz od 7230 do 7235
- Założenia:
 - czytelność (protokół tekstowy)
 - klient - serwer
 - zapytanie - odpowiedź
- HTTP 2.0
 - binarny
 - mechanizm PUSH
- Metody
 - GET, POST, PUT, DELETE
 - OPTIONS, HEAD, TRACE, CONNECT, PATCH

[Docs] [txt|pdf] [draft-ietf-httpbi...] [Diff1] [Diff2] [Errata]

PROPOSED STANDARD
Errata Exist

Internet Engineering Task Force (IETF)
Request for Comments: 2616
Obsoletes: 2145, 2616
Updates: 817, 824
Category: Standards Track
ISSN: 2076-1721

Host: www.tutorialspoint.com

Content-Type: text/html; charset=UTF-8

Content-Length: length

Accept-Language: en-us

The Hypertext Transfer Protocol (HTTP) is a stateless application-level protocol for distributed, collaborative, hypertext information systems. This document provides an overview of HTTP architecture and its associated terminology, defines the "http" and "https" Uniform Resource Identifier (URI) schemes, defines the HTTP/1.1 message syntax and parsing requirements, and describes related security concerns for implementations.

license=string&content=string&/paramsXML=string

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in [Section 2 of RFC 5741](#).

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <http://www.rfc-editor.org/info/rfc7230>.

Technology/HTTP back-end	Overall Parsing Result	Example
ASP.NET/IIS	All occurrences of the specific parameter	par1=val1,val2
ASP/IIS	All occurrences of the specific parameter	par1=val1,val2
PHP/Apache	Last occurrence	par1=val2
PHP/Zeus	Last occurrence	par1=val2
JSP,Servlet/Apache Tomcat	First occurrence	par1=val1
JSP,Servlet/Oracle Application Server 10g	First occurrence	par1=val1
JSP,Servlet/Jetty	First occurrence	par1=val1
IBM Lotus Domino	Last occurrence	par1=val2
IBM HTTP Server	First occurrence	par1=val1
mod_perl,libapreq2/Apache	First occurrence	par1=val1
Perl CGI/Apache	First occurrence	par1=val1
mod_perl,lib???/Apache	Becomes an array	ARRAY(0x8b9059c)
mod_wsgi (Python)/Apache	First occurrence	par1=val1
Python/Zope	Becomes an array	['val1', 'val2']
IceWarp	Last occurrence	par1=val2
AXIS 2400	All occurrences of the specific parameter	par1=val1,val2
Linksys Wireless-G PTZ Internet Camera	Last occurrence	par1=val2
Ricoh Aficio 1022 Printer	First occurrence	par1=val1
webcamXP PRO	First occurrence	par1=val1
DBMan	All occurrences of the specific parameter	par1=val1~~val2

<https://dunnesec.com/category/attacks-defence/http-parameter-pollution/>

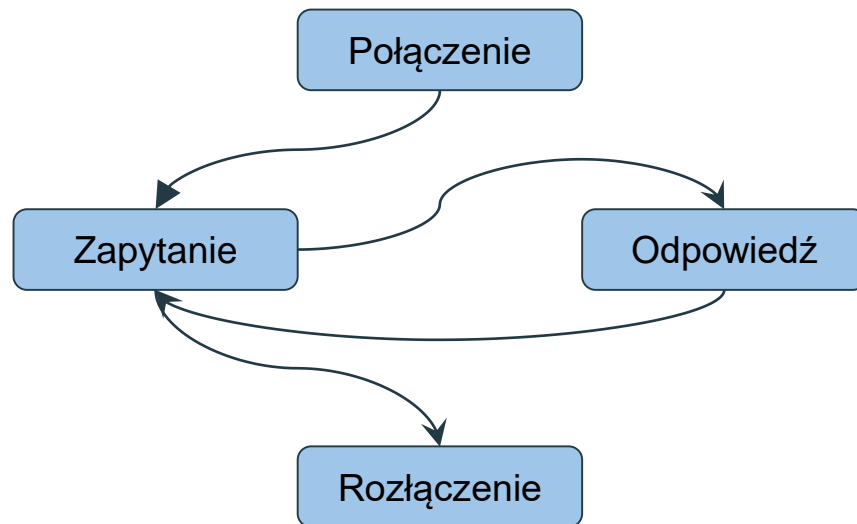
Pierwszy scenariusz ataku

- URL: http://voting.com/poll.jsp?poll_id=2536
 - Kandydat 1: http://voting.com/vote.jsp?poll_id=2536&candidate_id=1
 - Kandydat 2: http://voting.com/vote.jsp?poll_id=2536&candidate_id=2
 - Kandydat 3: http://voting.com/vote.jsp?poll_id=2536&candidate_id=3
- Spreparowany URL: http://voting.com/poll.jsp?poll_id=2536%26candidate_id%3D2
 - Aplikacja dokleja parametr z każdym kandydatem.
 - Kandydat 1: http://voting.com/vote.jsp?poll_id=2536&candidate_id=2&candidate_id=1
 - Kandydat 2: http://voting.com/vote.jsp?poll_id=2536&candidate_id=2&candidate_id=2
 - Kandydat 3: http://voting.com/vote.jsp?poll_id=2536&candidate_id=2&candidate_id=3

Pierwsza ochrona przed atakiem

- HPP - HTTP Parameter Tampering (nie ma w OWASP Top 10)
- V5: Malicious input handling verification requirements
- Ochrona?
 - Weź pod rozwagę możliwość wystąpienia wielu parametrów.
 - Określ dokładne wyrażenia regularne dla przekierowań.
 - Używaj kodowania URL, żeby nie wklejać zdekodowanych parametrów w treść strony.

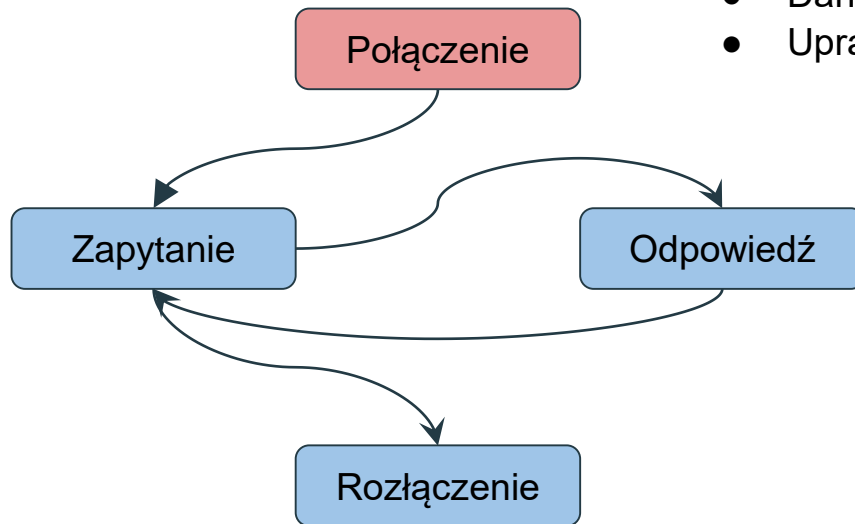
Baza danych



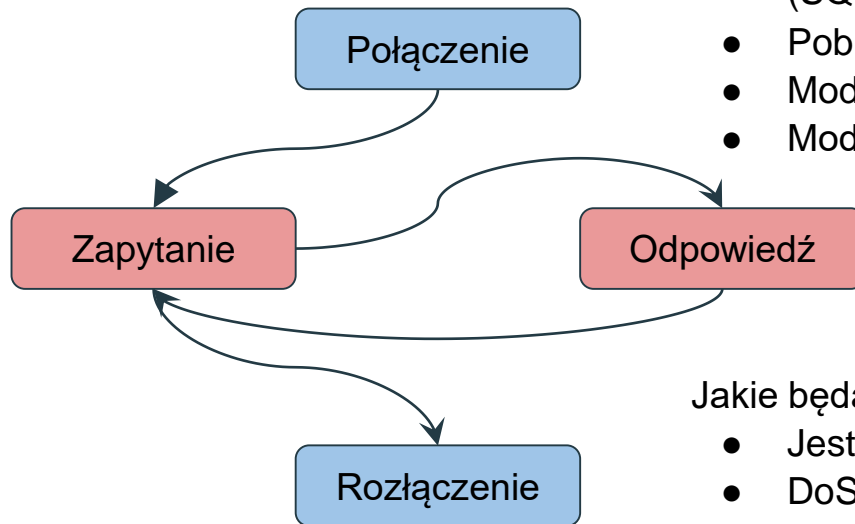
Baza danych

Co może pójść nie tak?

- Dostęp z zewnątrz.
- Dane logowania użytkownika.
- Uprawnienia użytkownika.



Baza danych



Co może pójść nie tak?

- Wstrzyknięcie złośliwego zapytania. (SQL Injection)
- Pobranie danych z bazy.
- Modyfikacja danych.
- Modyfikacja uprawnień.

Jakie będą konsekwencje?

- Jestem adminem.
- DoS
- RCE!

Baza danych - SQL

- Bazy relacyjne (np. MySQL, PostgreSQL, MsSQL).
- DQL (Data Query Language)

```
SELECT * FROM employees WHERE salary > 2000 ORDER BY age DESC;
```

- DML (Data Manipulation Language)

```
INSERT INTO employees (name, salary, age) VALUES ('John Brown', 2200, 30);
```

- DDL (Data Definition Language)

```
CREATE TABLE employees ( name VARCHAR(255), salary FLOAT, age INT);
```

- DCL (Data Control Language)

```
GRANT SELECT, INSERT, UPDATE ON employees TO webuser;
```

Baza danych - SQL Injection

- A1 – Injection
- V5. Malicious input handling
- Błędy implementacji:
 - Sklejki zapytania SQL.
 - Wklejanie danych użytkownika bezpośrednio do zapytania SQL.



Baza danych - SQL Injection - Przykłady

- Załóżmy, że aplikacja wyświetla nazwę użytkownika.

```
String query = "SELECT * FROM user_data WHERE login = 'admin' AND password = 'cffcd2919d9c8ef793ce1ac07a440eda';  
SELECT * FROM user_data WHERE login = 'admin' AND password = 'cffcd2919d9c8ef793ce1ac07a440eda';
```

```
SELECT * FROM user_data WHERE login = 'admin' --  
SELECT * FROM user_data WHERE login = 'admin' --
```

```
# Remote code execution (MSSQL)
```

```
SELECT * FROM user_data WHERE login = '' --  
'; master.dbo.xp_cmdshell 'cmd.exe dir c:';--  
SELECT * FROM user_data WHERE login = '' --  
'cmd.exe dir c:';--' AND password = '...';  
SELECT * FROM user_data WHERE login = '' --  
(NULL, 'hack', '<hash>', 1) --  
(NULL, 'hack', '<hash>', 1) --' AND password = '...';
```

Baza danych - Blind SQL Injection

- Załóżmy, że aplikacja nie wyświetla nazwy użytkownika, a jedynie informuje, czy zapytanie zwraca dane, czy odpowiedź jest pusta.

```
String query = "SELECT * FROM user_data WHERE login = '" +  
request.getParameter("login") + "' AND MD5(password) = '" + passwordHashed  
+ "'";
```

```
admin' AND substring(password,1,1) = 'a' --  
SELECT * FROM user_data WHERE login = 'admin' AND substring(password,1,1) =  
'a' -- ' AND password = '...';  
SELECT * FROM user_data WHERE login = 'admin' AND substring(password,1,1) =  
'a'
```

Baza danych - SQL Injection - Historie

- Sony Pictures (2011)
 - Wykradziono:
 - dane użytkowników,
 - 3,5 mln kuponów,
 - 700 tys. kodów muzycznych.
 - <http://www.macnn.com/articles/11/06/02/lulz.security.hits.sony.again.in.security.message/>
- Talk Talk (2015)
 - Wykradziono dane 156959 klientów.
 - I jeszcze kara 400 tys. funtów
 - <https://ico.org.uk/about-the-ico/news-and-events/news-and-blogs/2016/10/talktalk-gets-record-400-000-fine-for-failing-to-prevent-october-2015-attack/>
- Rosyjska Ambasada w Armenii (2016)
 - Wykradzione dane logowania administratora.
 - <http://securityaffairs.co/wordpress/54393/hacking/russian-embassy-of-armenia-hacked.html>
- Wordpress (głównie pluginy), Joomla, Moodle i naprawdę dużo innych...

Baza danych - SQL Injection - Ochrona

- Ochrona przed SQL Injection:
 - Wykorzystanie mechanizmu Prepared Statements (with Parameterized Queries).
 - Wykorzystanie mechanizmu Stored Procedures.
 - Czyszczenie (escaping) danych przed przekazaniem do zapytania.
 - Whitelisting.

```
# OWASP Enterprise Security API (ESAPI)
# https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API
@param = "text with a quote" INTO REGEXP_REPLACE (password, '"/', ' ');
@param = mysql_real_escape_string($char);
Encoder oe = new OracleEncoder();
String query = "SELECT user_id FROM user_data WHERE user_name = '" + oe.encode(req.getParameter("userName")) + "' and user_password = '" + oe.encode(req.getParameter("pwd")) + "'";
try {
    ResultSet rs = stmt.executeQuery(query);
    while (rs.next()) {
        // logging and error handling
    }
}
```

Baza danych - Frameworki

- Framework (np. Hibernate, Doctrine, Yii Active Record) nie chroni przed SQLi.

```
Query unsafeHQLQuery = session.createQuery("from Inventory where  
productID='"+userSuppliedParameter+"'");
```

```
Query safeHQLQuery = session.createQuery("from Inventory where  
productID=:productid");  
safeHQLQuery.setParameter("productid", userSuppliedParameter);
```

- Dostęp do danych przez model frameworkowy jest zwykle bezpieczny.

```
$post=new Post;  
$post->title='przykładowy post';  
$post->content='zawartość postu';  
$post->save();
```


Nierelacyjne bazy danych

- Bazy nierelacyjne - MongoDB, CouchDB, Redis i bardzo dużo innych.
- Czy SQL Injection jest w nierelacyjnych bazach?
- Nie, ale...
jest NoSQL Injection :)
- Język zapytań w MongoDB - BSON (Binary JSON)
 - Wstrzyknięcia występują, gdy zapytania przyjmują wyrażenia w Javascript
 - \$where - Use the \$where operator to pass either a string containing a JavaScript expression or a full JavaScript function to the query system.

```
db.myCollection.find( { active: true, $where: "this.credits - this.debits < 0" } );  
db.myCollection.find( { active: true, $where: function() { return obj.credits - obj.debits <  
0; } } } );
```

- **Column:** Accumulo, Cassandra, Druid, HBase, Verica, SAP HANA
- **Document:** Apache CouchDB, ArangoDB, Clusterpoint, Couchbase, Cosmos DB, HyperDex, IBM Domino, MarkLogic, MongoDB, OrientDB, Qiz
- **Key-value:** Aerospike, ArangoDB, Couchbase, Dynamo, FairCom c-treeACE, FoundationDB, HyperDex, InfinityDB, MemcacheDB, MUMPS, Or
- **Graph:** AllegroGraph, ArangoDB, InfiniteGraph, Apache Giraph, MarkLogic, Neo4J, OrientDB, Virtuoso
- **Multi-model:** Alchemy Database, ArangoDB, CortexDB, Couchbase, FoundationDB, InfinityDB, MarkLogic, OrientDB

Nierelacyjne bazy danych - NoSQLi

```
db.myCollection.find( { active: true, $where: function() { return  
obj.credits - obj.debits < $userInput; } } );
```

- Jak zweryfikować, czy jest podatne na wstrzyknięcia?
 - Sprawdzić, czy nie ma błędu, gdy przekazane zostaną specjalne znaki: '"\;{}

```
(function(){var date = new Date(); do{curDate = new Date();}while(curDate-date<10000);  
return Math.max();})();
```

```
db.myCollection.find( { active: true, $where: function() { return obj.credits - obj.debits <  
(function(){var date = new Date(); do{curDate = new Date();}while(curDate-date<10000);  
return Math.max();})(); } } );
```

MongoDB - NoSQLi

```
db.collection('users').find({ "user": req.query.user, "password":  
req.query.password });
```

```
https://example.org/login?user=patrick&password[%24ne]=
```

```
db.collection('users').find({ "user": "patrick", "password": { "&ne": "" }  
});
```

Redis - NoSQLi

```
RedisClient.expireat(  
    req.query.key,  
    new Date("November 8, 2026 11:13:00").getTime()  
);
```

```
https://example.org/expire?key[]=foo&key[]=1117542887
```

```
RedisClient.expireat(  
    "foo",  
    1117542887  
);
```

NoSQL - Jak zabezpieczyć?

- Weryfikacja typu danych.
 - {'password': 'PASS'} vs {'password': {'&ne': ''}}
- Ograniczenie danych wejściowych od użytkownika do prostych typów.
 - Integer
 - Napis
- Weryfikacja dostępnych wartości danych (np. whitelist, o ile możliwe).