IT/Dev CONNECTIONS

WEB APPLICATION SECURITY TODAY

Christian Wenz
CEO
Actition GmbH

Delivered by KNect365 TMT an Informa business

1. INJECTION
2. BROKEN AUTHENTICATION
3. CROSS-SITE SCRIPTING (XSS)
4. BROKEN ACCESS CONTROL
5. SECURITY MISCONFIGURATION
6. SENSITIVE DATA EXPOSURE
7. INSUFFICIENT ATTACK PROTECTION
8. CROSS-SITE REQUEST FORGERY (CSRF)
9. USING COMPONENTS WITH KNOWN VULNERABILITIES
10. UNDERPROTECTED APIS

1. INJECTION
2. BROKEN AUTHENTICATION
3. SENSITIVE DATA EXPOSURE
4. ~~XML EXTERNAL ENTITIES (XEE)~~
5. BROKEN ACCESS CONTROL
6. SECURITY MISCONFIGURATION
7. CROSS-SITE SCRIPTING
8. ~~INSECURE DESERIALIZATION~~
9. USING COMPONENTS WITH KNOWN VULNERABILITIES
10. INSUFFICIENT LOGGING AND MONITORING

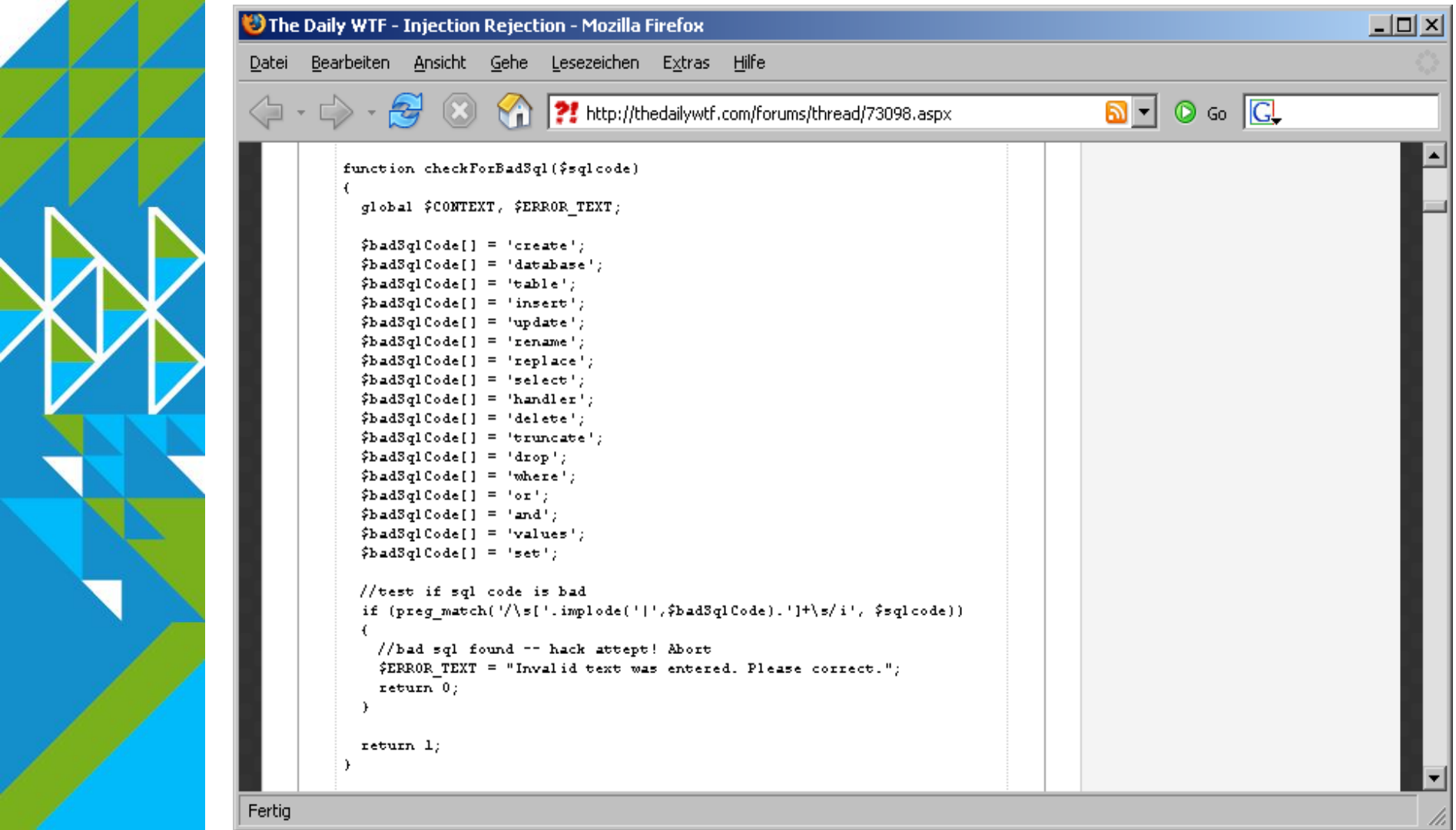
- Different kinds of injections

  - SQL Injection
  - LDAP Injection
  - XPath Injection
  - Regex Injection

- We are actually talking about SQL Injection

- OR Mappers help. Parametrized queries/prepared statements, too. If you must, database-specific escaping APIs.

- Basically safe, since no SQL is written (YMMV)
- Mind return values: IEnumerable<T> statt IQueryable<T>
- Avoid Database.SqlQuery<TElement> (EF5+)!
  - Or at least use parameters
- More information: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/security-considerations


The Daily WTF - Injection Rejection - Mozilla Firefox
Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe
http://thedailywtf.com/forums/thread/73098.aspx
Go
function checkForBadSql($sqlcode)
{
global $CONTEXT, $ERROR_TEXT;
$badSqlCode[] = 'create';
$badSqlCode[] = 'database';
$badSqlCode[] = 'table';
$badSqlCode[] = 'insert';
$badSqlCode[] = 'update';
$badSqlCode[] = 'rename';
$badSqlCode[] = 'replace';
$badSqlCode[] = 'select';
$badSqlCode[] = 'handler';
$badSqlCode[] = 'delete';
$badSqlCode[] = 'truncate';
$badSqlCode[] = 'drop';
$badSqlCode[] = 'where';
$badSqlCode[] = 'or';
$badSqlCode[] = 'and';
$badSqlCode[] = 'values';
$badSqlCode[] = 'set';
//test if sql code is bad
if (preg_match('/\s['.implode('|',$badSqlCode).']+\s/i', $sqlcode))
{
//bad sql found -- hack attept! Abort
$ERROR_TEXT = "Invalid text was entered. Please correct.";
return 0;
}
return 1;
}
Fertig

- HTTP is a stateless protocol
- Session management is, essentially, a hack
- Different attack vectors
  - Session hijacking
  - Session fixation
  - Other issues: insufficient session timeout, unencrypted passwords, …

- Use encryption throughout
- HTTPS only (if possible, enforce it)
  - HTTP Strict Transport Security
  - „secure“ cookie flag

- 2013 list: Insecure Direct Object References & Missing Function Level Access Control
- Access control to data
- Access control to functions

haslo
@haslo

Follow

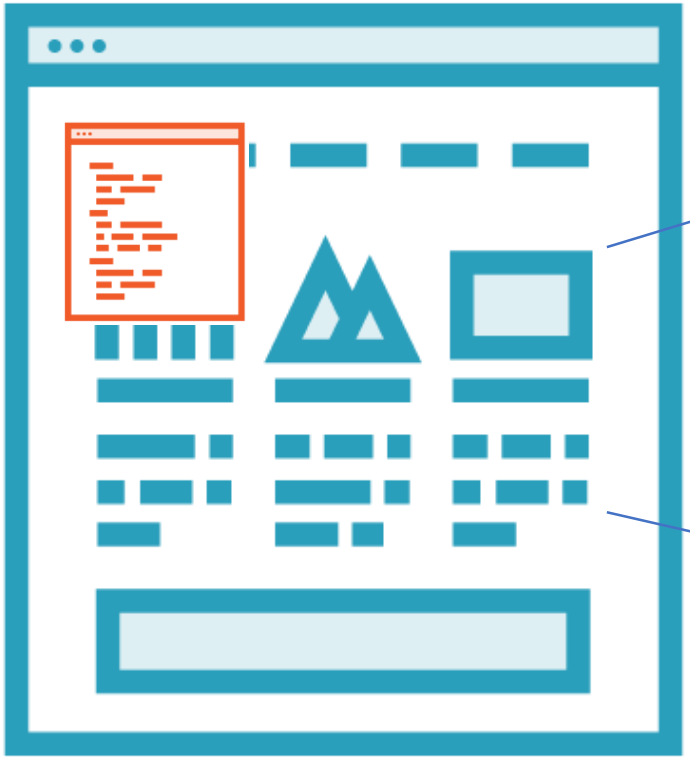
Twitter enabled 280 characters for a select few. I'm not part of that group. I'm absolutely certain that them implementing such restrictions in the client only and not on the server is a great idea, and I happen to express that certainty with a tweet that is 280 characters long.

11:31 AM - 27 Sep 2017

- I´m not an admin!
- In the days of DevOps, maybe I am
- Harden server/OS
- Do not send detailled error messages to the client
- Use browser security headers

- One of the oldest attacks around
- Injecting content (usually JavaScript) into server output
- Might bring friends: if there is XSS, it‘s very hard to defend against CSRF

- DOM-based (type 0): Everything happens in the client
- Persistent (type 1): Server stores malicious input
- Non-persistent (type 2): Server echoes back malicious input



http://example.com:80/page.php

http://example.com/script.js

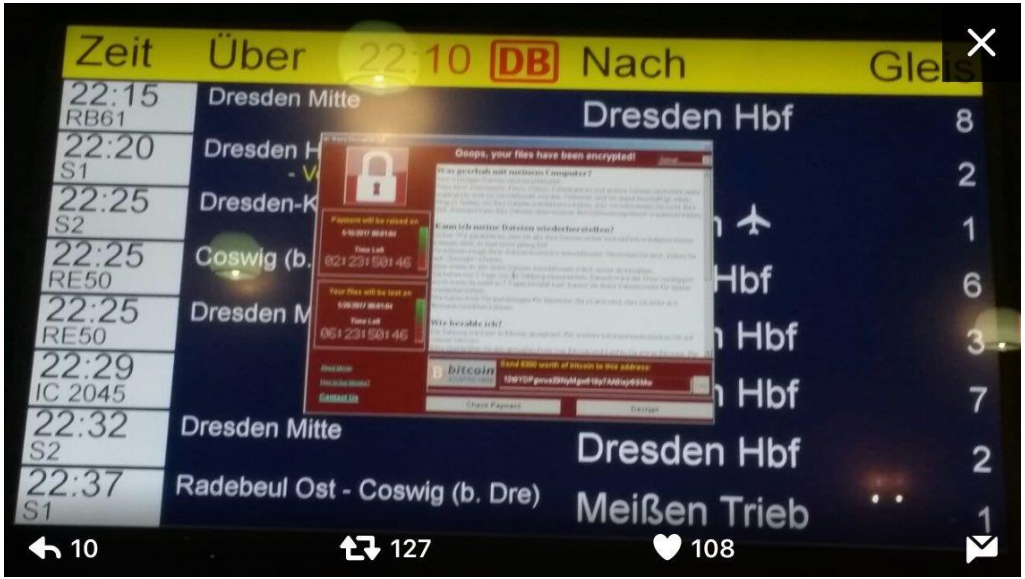
https://code.jquery.com/
jquery-x.y.z.min.js

Protocol

Domain

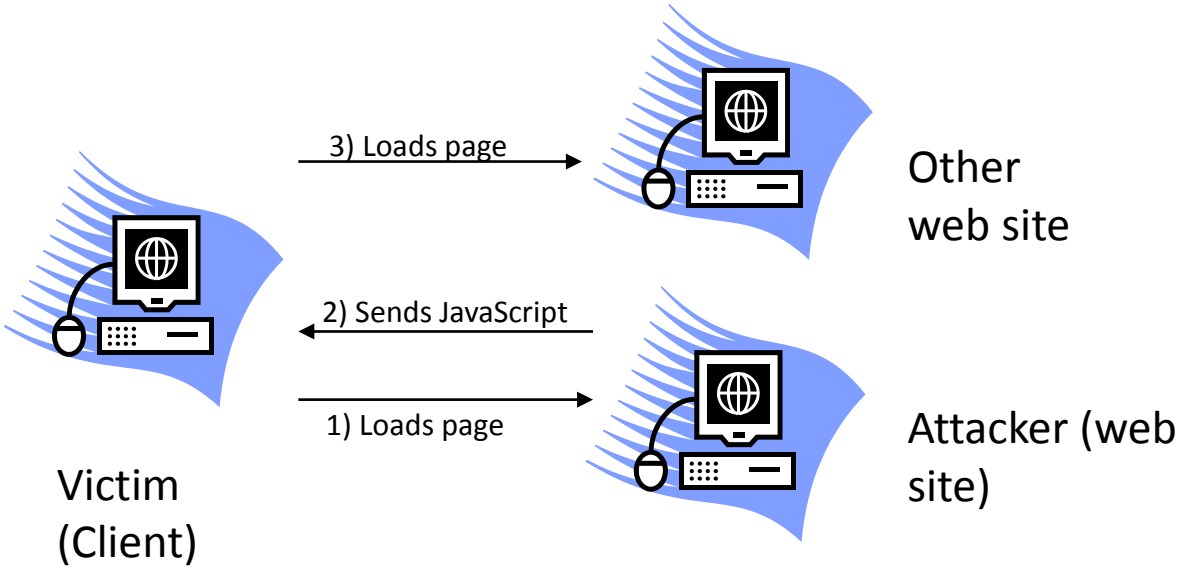
Port

- Escape output (< > " ' &)
- Use browser protection (X-XSS-Protection)
- Use Content Security Policy



https://twitter.com/jacobhusted69/status/863151760118083584/photo/1

- Do log!
- Do look at the logs!


3) Loads page
Other
web site
2) Sends JavaScript
1) Loads page
Attacker (web
site)
Victim
(Client)

- Unvalidated Redirects and Forwards (#10 in the 2013 list)
- Accessing opener
- Regex DoS
- … and more! (Oct 18, 10:15 am)

- Thank you!
- info@christianwenz.de
- @chwenz