

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Developer and IT Training

Injection

OWASP Web Security Threat #3

25°C Mostly cloudy

Search

1

6:38 PM

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Home > Web Security Fundamentals Techniques > Injection

u2u

Agenda

- Introduction
- Visualizing the attack
- Injection defenses
- Never trust user input
- Web-specific canonicalization
- User input remedies
- Password policies

25°C Mostly cloudy

Search

2 6:38 PM 12/29/2023

The screenshot shows a web browser window with the following details:

- Title Bar:** u2u U2U Online
- Address Bar:** online.u2u.be/courses/25896/modules/11/24857
- Page Content:**
 - Section Header:** #3: Injection
 - List:**
 - What is an injection attack?
 - Exploiting any kind of user input to inject code so it gets executed directly on the server
 - The most known injection attacks are
 - SQL injection
 - LDAP injection
 - JavaScript injection
 - ... injection
 - Buffer overflow- Left Sidebar:** A navigation tree under the heading "Injection". It includes categories like "Injection", "Information", "Visualizing the attack", "Arbitrary Updates", "Cross-site Scripting", "Remote Code Execution", "Run with Local Privilege", "Use Undefined Variable", "Value from user input", "Web-specific Considerations", and "User Input Sanitization".
- Right Sidebar:** A large "u2u" logo.
- Bottom Taskbar:** Shows the Windows Start button, a search bar, pinned icons for File Explorer, Task View, Control Panel, Internet Explorer, Google Chrome, and others, and system status icons for battery, signal, and date/time (6:38 PM, 12/29/2023).

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Home > Web Security Fundamentals Techniques > Injection

Injection - Risks

u2u

- Data loss or corruption
- Data could be stolen or manipulated
- Unauthorized access
- Denial of access
- Complete host system takeover
- Loss of reputation

25°C Mostly cloudy

Search

6:38 PM 12/29/2023

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Home > Web Security Fundamentals Techniques > Injection

Injection

SQL Injections

1. Direct

2. Indirect

3. Bypassing the attack

4. SQL injection

5. Cross-site Scripting

6. Header Overwrite

7. Run with Local Privilege

8. Use Dangerous Intensity

9. Value from user input

10. Web-specific Considerations

11. User Input Sanitization

Summary

Security Matrix

u2u

ATTACK VECTORS	SECURITY WEAKNESS	TECHNICAL IMPACTS
Exploitability	Prevalence	Detectability
EASY	COMMON	EASY
		Impact
		SEVERE

25°C Mostly cloudy

Search

6:38 PM 12/29/2023

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Injection

- Introduction
- Visualizing Database
- SQL Injection
- Code White Spacing
- Hexadecimal
- Run with Layout Message
- Use Inference Integrity
- Use Cross Site Scripting
- Web Specific Considerations
- User Input Sanitization
- Summary

Prevalence: COMMON

u2u

- Try following google query:

- Pick a site, and modify id= in url

www.bible-history.com/subcat.php?id='

Query failed : You have an error in your SQL syntax; check the r

25°C Mostly cloudy

6:38 PM 12/29/2023

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Home > Web Security Fundamentals Techniques > Injection

Injection

- Injection
- Infiltration
- Visualizing the attack
- Advanced injection
- Code injection
- Reflected XSS
- DOM-based XSS
- Reflected SQL injection
- Blind SQL injection
- Use undefined property
- Value of null user input
- Web-specific connections
- User input terminals

Summary

Visualizing the attack

u2u

The diagram illustrates a SQL injection attack flow between a user and a database server.

1. A stick figure on the left sends a **Malicious request** (red arrow) to a server rack icon in the middle.

2. The server rack icon sends a **Response with extra data** (red arrow) back to the stick figure.

3. The server rack icon sends a **Executes modified query** (red arrow) to a database cylinder icon on the right.

4. The database cylinder icon sends a **Returns result from modified query** (red arrow) back to the server rack icon.

25°C Mostly cloudy 6:38 PM 12/29/2023

U2U Online

online.u2u.be/courses/25896/modules/11/24857

Injection

- Injection
- SQL Injection

SQL Injection

<https://www.youtube.com/watch?v=jKyIhJtPmI>

u2u

- Let's consider the following SQL command

```
"SELECT * FROM Users WHERE u.UserName=' " + username + " ' AND u.Password=' " + pw + " '"
```

- The idea is to select a user entry matching a username
 - So a command could be turned into

```
"SELECT * FROM Users WHERE u.UserName='Diedrik' AND u.Password='123456'
```

Login here:

Diedrik

123456

Log in

25°C Mostly cloudy

Search

6:38 PM 12/29/2023

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Home > Web Security Fundamentals Techniques > Injection

SQL Injection

u2u

- What could happen if our user was less cooperative?
 - Our user could decide to type in something a bit more elusive:

Login here:

Diedrik

' OR 1=1; --

Log in

- The statement now becomes

"SELECT * FROM Users WHERE u.UserName='Diedrik' AND u.Password='OR 1=1; --'

25°C Mostly cloudy

Search

6:38 PM 12/29/2023

The screenshot shows a web browser window titled "U2U Online" with the URL "online.u2u.be/courses/25896/modules/11/24857". The main content area displays a slide titled "SQL Injection" with a red header bar. The slide contains two bullet points explaining how a SQL injection attack works:

- That statement will look up a user with a certain username, combined with a certain password but also when 1 equals 1, which is always...
 - In other words, we'll get a user back in any case
- Because this user object from the database is then stored as credentials for our malicious client, we'll be logged in as that user

Below the slide, a dark rectangular box displays the text "Welcome Diedrik". The browser's status bar at the bottom shows the weather (25°C, Mostly cloudy), system icons (Search, File Explorer, Task View, etc.), and the date/time (6:38 PM, 12/29/2023).

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Home > Web Security Fundamentals Techniques > Injection

SQL Injection

u2u

- Now our current malicious user is still pretty friendly
 - He could decide to play around with the website a bit
- Other things a user could execute
 - DROP database table
 - INSERT new information
 - ...

Login here:

Lieven

':; INSERT INTO [4U2].[Users]([UserName],[Description],[UserRole],[Password]) VALUES ('Mallory','Super Admin',1,'1337 H4X0R') --

Log in

Lieven

'; DROP TABLE Users; --

Log in

25°C Mostly cloudy

Search

6:38 PM 12/29/2023

The screenshot shows a web browser window with the following details:

- Title Bar:** u2u U2U Online
- Address Bar:** online.u2u.be/courses/25896/modules/11/24857
- Left Sidebar:** A navigation menu for "Injection" techniques, including "Reflection", "Environment", "Cross-site Scripting", "Remote Code Execution", "Run with Local Privilege", "Use Database Integrity", "Value from user input", "Web-specific Considerations", and "User Input Sanitization".
- Content Area:**
 - ## Sources of untrusted data
 - From the **user**
 - In the URL, either a query string, or from the route
 - POSTed from the form
 - From the **browser**
 - In cookies
 - From a header
 - From other locations
 - From an external **service**
 - From your own **database!**- Bottom Status Bar:** 25°C Mostly cloudy, Search bar, Taskbar icons (File Explorer, Edge, File Manager, Google Chrome), and system status indicators (Battery, Wi-Fi, Volume, Language: ENG, Date: 12/29/2023).

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Home > Web Security Fundamentals Techniques > Injection

Injection

- Introduction
- SQL Injections
- Blind SQL Injections
- Boolean Blind SQL Injections
- Time-based SQL Injections
- Oracle Database
- PostgreSQL Database
- MySQL Database Integrity
- Oracle Database Integrity
- Web-specific Considerations
- User Input Sanitization

Summary

u2u

25°C Mostly cloudy

Search

6:38 PM 12/29/2023

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/11/24857. The page content is as follows:

SQL Injection example from the real world

TIME

SONY

New Sony Hack Claims Over a Million User Passwords

By Doug Aamoth | June 02, 2011

Share: Facebook Like 2 Twitter Tweet Google+ 0 LinkedIn Share 321 Pinterest Pin it Read Later

Another of Sony's websites has reportedly been hacked—this time around, the victim is [SonyPictures.com](#). The group claiming responsibility for the breach, "LulzSec," is the same group behind the [recent PBS website hack](#).

A statement from the group reads, in part:

"SonyPictures.com was owned by a very simple SQL injection, one of the most primitive and common vulnerabilities, as we should all know by now. From a single injection, we accessed EVERYTHING. Why do you put such faith in a company that allows itself to become open to these simple attacks? What's worse is that every bit of data we took wasn't encrypted. Sony stored over 1,000,000 passwords of its customers in plaintext, which means it's just a matter of taking it. This is disgraceful and insecure: they were asking for it."

Follow @techland

25°C Mostly cloudy 6:38 PM 12/29/2023

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Developer and IT Training

Demo

SQL Injection

25°C Mostly cloudy

Search

6:38 PM

15

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/11/24857. The page content is a slide titled "Developer and IT Training" with a large red "Demo" heading. Below the heading is the sub-section "SQL Injection". On the left side, there is a sidebar with a navigation tree under the "Injection" category, which includes "Injection", "Injection", "SQL Injection", "Cross-site Scripting", "Reflected XSS", "DOM-based XSS", "Blind XSS", "Use Dangerous Integers", "Arbitrary File Upload", "Web-specific Considerations", and "User Input Sanitization". A summary section is also present. The top right corner features the "u2u" logo. The bottom of the screen shows the Windows taskbar with various pinned icons and the system tray indicating the date and time.

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Injection

- ↳ Injection
- ↳ Cross-Site Scripting
- ↳ SQL Injection
- ↳ Clickjacking
- ↳ DOM-based XSS
- ↳ Cross-Site Scripting
- ↳ Reflected XSS
- ↳ Persistent XSS
- ↳ User-Defined Integrity
- ↳ Value of an XSS
- ↳ Web-specific Considerations
- ↳ User Input Sanitization
- ↳ Summary

JavaScript Injection

u2u

- Does your site use external JavaScript libraries?
 - Then you're open for JavaScript injection
- For example, a website's payment page uses third party JavaScript
 - A hacker sees this and modifies the JavaScript to steal data from the page (Credit Card)

25°C Mostly cloudy

Search

6:38 PM 12/29/2023

Real-life Examples

Radisson Hotel Group suffers data breach, customer info leaked

Radisson Hotel Group loyalty scheme members are affected and may have had their personal information stolen.

Card-stealing code that pwned British Airways, Ticketmaster pops up on more sites via hacked JS

Feedify's whack-a-mole with MageCart malware miscreants

Cross Site Scripting

- XSS or Cross Site Scripting
 - Application takes untrusted data and sends it to a web browser without sanitization
 - Allows attackers to execute scripts on the victim's browser
- Famous examples
 - TweetDeck (Twitter client) had a problem with a self retweeting tweet
(<https://www.youtube.com/watch?v=zv0kZKC6GAM>)
 - StrongWebmail CEO's mail account hacked via XSS
 - ...

25°C Mostly cloudy 6:38 PM ENG 12/29/2023

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/11/24857. The main content area displays a slide titled "XSS attack" with a red horizontal bar below it. The slide contains two bullet points:

- The attack exists where there are fields that are not sanitized correctly
- Different types:
 - Non-Persistent XSS (e.g.: showing the query from the querystring on screen)
 - Persistent XSS (e.g.: storing a comment in the database)
 - DOM-based XSS (attack takes place entirely inside the browser)
 - mXSS (or mutation XSS)

The browser interface includes a sidebar on the left with a navigation tree under "Injection" and a "u2u Online" status bar at the bottom. The taskbar at the bottom of the screen shows various application icons and system status.

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

In Practice

■ Take some user input
– And show user input on screen

CommentText <Script>alert('hello')</Script>

Create

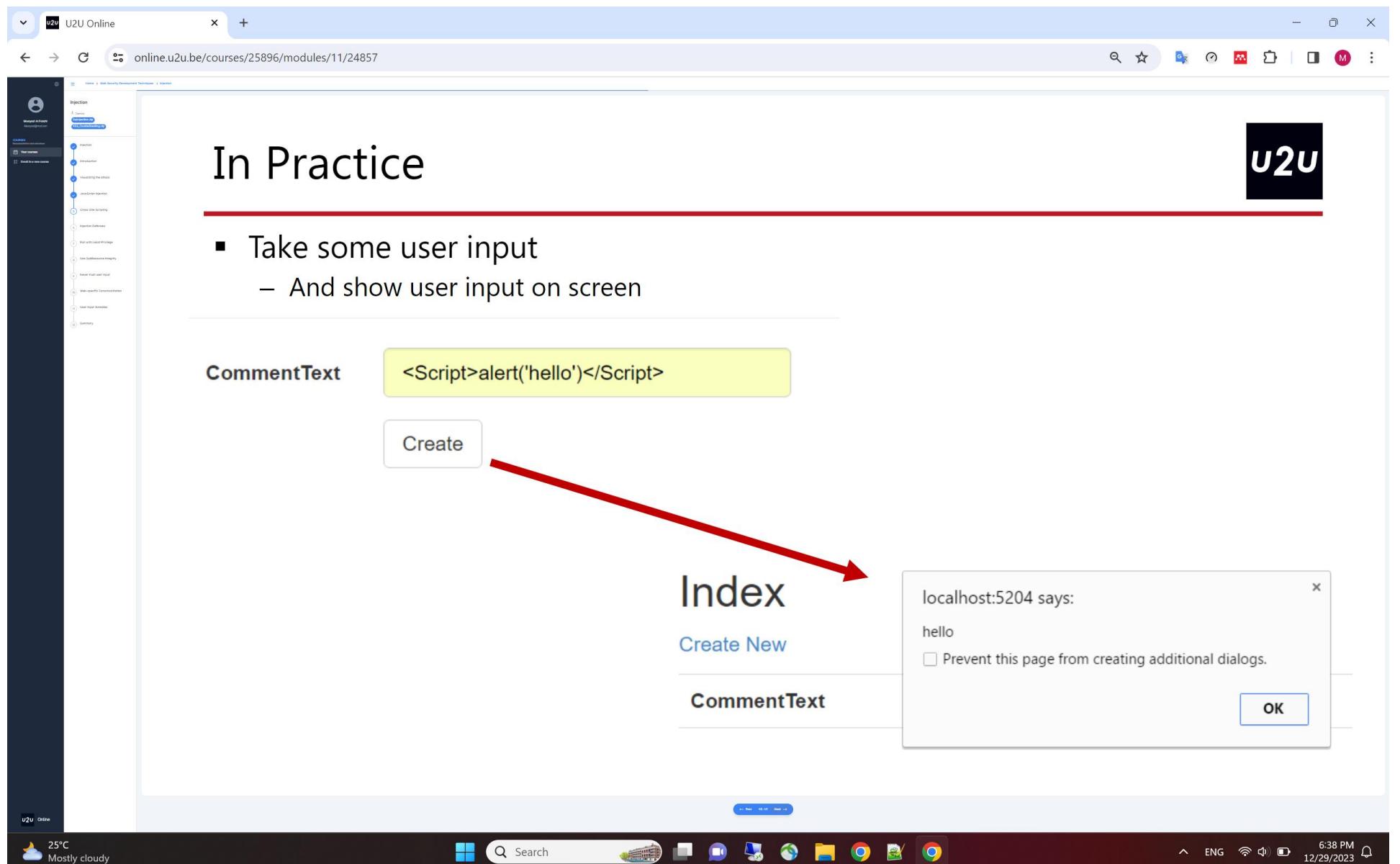
Index

Create New

CommentText

localhost:5204 says:
hello
 Prevent this page from creating additional dialogs.

OK



25°C Mostly cloudy 6:38 PM 12/29/2023

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/11/24857. The main content area displays a slide titled "XSS - Feasability" with the following bullet points:

- Chrome (and others) stop non-persistent XSS
 - Support varies on different browsers (so you can't rely on it!!!)

Below the slide, the browser's developer tools are open, specifically the "Console" tab. The log shows a single entry in red text:

```
✖ The XSS Auditor refused to execute a script in 'http://localhost:5204/Home/XSS' because its source      XSS:44  
code was found within the request. The auditor was enabled as the server sent neither an 'X-XSS-Protection'  
nor 'Content-Security-Policy' header.
```

At the bottom of the slide, there is a link: https://www.owasp.org/index.php/OWASP_Secure_Headers_Project.

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Home > Web Security Fundamentals Techniques > Injection

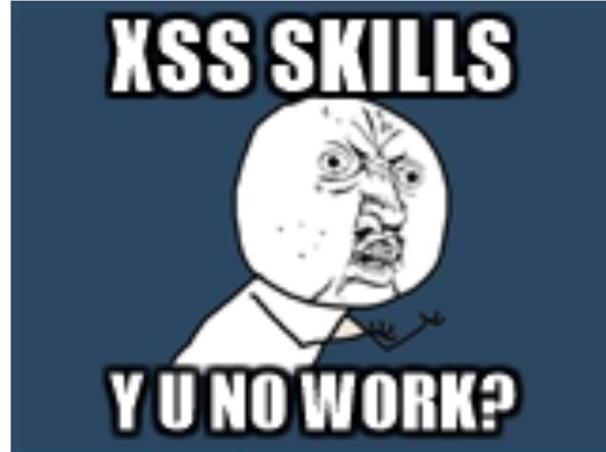
Injection

- Reflective
- DOM-based
- CSS-injection
- JavaScript injection
- Cross-Site Scripting
- Header Hijacking
- Reflected XSS
- User-Defined Integrity
- User Input Validation
- Web-specific Considerations
- User Input Sanitization

u2u

XSS - Feasability

- XSS could work
 - On a website that needs to take in raw HTML like an e-mail host or CMS system
 - On a website that needs to show HTML coming from the user like a rich comment section



25°C Mostly cloudy

Search

6:38 PM 12/29/2023

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Home > Web Security Fundamentals Techniques > Injection

SQL Injection - Defenses



- Validate the input coming in and sanitize it
 - Whitelist data
- Escape the characters that are being used for SQL Injection attacks
 - e.g.: ‘ (single quote)
 - Most languages have an escape method
- Use parameterized SQL Statements - .NET supports this OOB
 - ADO.NET
 - Entity Framework
- Change Database permissions (run with minimal permissions)

25°C Mostly cloudy

Search

6:38 PM 12/29/2023

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Injection

- Injection
- SQL Injection
- Blind SQL Injection
- Parameterized Queries
- Prepared Statements
- Common Table Expressions
- Row-Level Security
- Use Default/Current Identity
- Value from user input
- Web-specific Connections
- User Input Sanitization

Parameterized Sql Statement (ADO.NET)

C#

```
var command = client.CreateCommand();

command.CommandText = @"SELECT * FROM [4U2].[Users] u
    INNER JOIN [4U2].[Roles] r
    ON u.UserRole = r.RoleID
    WHERE u.UserName=@username AND u.Password=@password;";

command.Parameters.AddWithValue("username", username);
command.Parameters.AddWithValue("password", password);

var reader = command.ExecuteReader();
```

25°C Mostly cloudy

Search

6:38 PM 12/29/2023

Building SQL Statements safely

- Use Stored Procedures
 - But you can still build vulnerable stored procedures!
- Use parameterized queries
 - Automatic with **Linq to SQL** and **Entity Framework**
- Make parameters strongly typed

25°C Mostly cloudy Search 6:38 PM

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/11/24857. The slide title is "SQL Injection in Entity Framework Core". On the left, there's a sidebar titled "Injection" with a tree view of various injection techniques. The main content area contains a bulleted list and code snippets.

- Normally, Entity Framework Core parameterizes all your queries
 - Except when using **FromSqlRaw**

```
context.Courses.FromSqlRaw(  
    $"SELECT CourseId, Title, Date FROM Courses WHERE CourseId = {id}"  
);
```

- Instead, use **FromSqlInterpolated**

```
context.Courses.FromSqlInterpolated(  
    $"SELECT CourseId, Title, Date FROM Courses WHERE CourseId = {id}"  
);
```

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Developer and IT Training

Demo

Using Stored Procedures

25°C Mostly cloudy

Search

6:38 PM

u2u Online

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/11/24857. The page title is "Change Database Permissions". On the left, there's a sidebar with a navigation tree under "Injection" and a "Summary" section. The main content area contains a bulleted list and a SQL command. The bottom of the screen shows a taskbar with various icons and system status.

Change Database Permissions

- Give accounts just enough privileges
 - E.g., external users should only be given read rights on the products table

```
DENY INSERT, UPDATE, DELETE ON PRODUCTS TO Guest
```

25°C Mostly cloudy Search 6:38 PM

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Home > Web Security Fundamentals Techniques > Injection

Injection

- SQL Injection
- Blind SQL Injection
- Arbitrary Selection
- Arbitrary Insertion
- Arbitrary Update
- Arbitrary Deletion
- Blind XSS
- SQL Injection Integrity
- Arbitrary User Input
- Web-specific Constructions
- User Input Sanitization

Summary

Bobby Tables – Credits to xkcd

The comic strip consists of four panels:

- Panel 1:** A stick figure stands next to a small table, holding a piece of paper. The text on the paper reads: "HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE."
- Panel 2:** The same stick figure is shown again, with a thought bubble above them containing the text: "OH, DEAR - DID HE BREAK SOMETHING? IN A WAY -".
- Panel 3:** The stick figure is shown again, with a thought bubble above them containing the text: "DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?". Below this, another thought bubble says: "OH, YES. LITTLE BOBBY TABLES, WE CALL HIM."
- Panel 4:** The stick figure is shown again, with a thought bubble above them containing the text: "WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY." Below this, another thought bubble says: "AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS."

u2u Online

25°C Mostly cloudy

Search

6:38 PM 12/29/2023

Use SubResource Integrity for JavaScript

- **SubResource Integrity** checks if the script is the one you want
 - Adds a hash to check against changes

```
<script src="https://code.jquery.com/jquery-1.10.2.min.js"
       integrity="sha256-
C6CB9UYIS9UJeqinPHWTHVqh/E1uhG5Twh+Y5qFQmYg="
       crossorigin="anonymous"></script>
```

- Can also be used for stylesheets

```
<link rel="stylesheet" href="https://site53.example.net/style.css"
      integrity="sha256-
vjnUh7+rXHH2lg/5vDY8032ftNVCIEC21vL6szrVw9M="
      crossorigin="anonymous">
```

u2u Online

online.u2u.be/courses/25896/modules/11/24857

Example of using SRI

- If you're using a CDN, most provide an SRI link for you
- For example: <https://code.jquery.com/>

Code Integration

```
<script
  src="https://code.jquery.com/jquery-3.3.1.min.js"
  integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
  crossorigin="anonymous"></script>
```

The `integrity` and `crossorigin` attributes are used for [Subresource Integrity \(SRI\) checking](#). This allows browsers to ensure that resources hosted on third-party servers have not been tampered with. Use of SRI is recommended as a best-practice, whenever libraries are loaded from a third-party source. Read more at [srihash.org](#)

25°C Mostly cloudy 6:38 PM 12/29/2023

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/11/24857. The page title is "Generating the SRI hash". On the left, there's a sidebar with a navigation tree under "Injection" and a "Summary" section. A large "u2u" logo is in the top right. The main content area has a red header bar with the text "Generating the SRI hash". Below it, a bullet point says "You can use the SRI Hash Generator at <https://www.srihash.org/>". The main content area has a large heading "SRI Hash Generator" and a form where the URL `https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.22.2/moment-with-locales.js` is entered. A blue button labeled "Hash!" is next to the input field. The resulting SRI hash is displayed in a dark box below: `<script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.22.2/moment-with-locales.js" integrity="sha384-5zARgXuvMiKGWrda+raHX0lTw717huPS47iQABU10DXHGQJ+7PE/PPghBZSmzaNU" crossorigin="anonymous"></script>`

Generating the SRI hash

▪ You can use the SRI Hash Generator at <https://www.srihash.org/>

SRI Hash Generator

Enter the URL of the resource you wish to use:

`https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.22.2/moment-with-locales.js`

Hash!

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/moment.js/2.22.2/moment-with-locales.js" integrity="sha384-5zARgXuvMiKGWrda+raHX0lTw717huPS47iQABU10DXHGQJ+7PE/PPghBZSmzaNU" crossorigin="anonymous"></script>
```

What is the **crossorigin** attribute?

- Loading scripts and stylesheets from other domains is not allowed
 - So, this requires CORS
- The crossorigin attribute tells the browser to fetch the file
 - In a way that allows reading it afterwards
 - Or to fail if CORS is not supported

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Home > Web Security Fundamentals Techniques > Injection

Injection

- SQL Injection
- XSS (Cross Site Scripting)
- Clickjacking
- Information Disclosure
- Denial of Service
- Cross Site Scripting
- Remote Code Execution
- Run with Least Privilege
- Use SafeResource Integrity
- Never trust user input
- Web-specific Considerations
- User Input Sanitization

Never Trust User Input!

u2u

- Be paranoid! Many people are out there to get you(r data)!
- Always validate input before you use it

All input is BAD until proven otherwise!

Data MUST be validated as it crosses the boundary between untrusted and trusted environments

25°C Mostly cloudy

Search

6:38 PM 12/29/2023

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/11/24857. The page title is "Common Excuse". On the left, there's a sidebar with a navigation tree under "Injection" and a "Your review" section. A large "u2u" logo is in the top right. The main content area contains two bullet points with sub-points:

- “Hey, I can’t check all the input, it will affect the application’s performance”
 - Performance is rarely a problem when checking user input!
- “Hey, I trust my users!”
 - Users can always make innocent mistakes which can still wreak havoc

The bottom of the screen shows a taskbar with various icons and system status indicators.

The screenshot shows a web browser window for 'U2U Online' at the URL online.u2u.be/courses/25896/modules/11/24857. The page title is 'Common Mistakes'. On the left, there's a sidebar with a navigation tree under 'Injection' and a 'Summary' section. The main content area contains a bulleted list of common mistakes:

- Trusting the user
 - Injection and XSS attacks
- Unbound sizes
 - Buffer overrun
- Using direct user input in SQL statements
 - SQL injection attack

The browser interface includes standard navigation buttons, a search bar, and a taskbar at the bottom with icons for weather, search, and various applications.

The screenshot shows a web browser window with the following details:

- Title Bar:** U2U Online
- Address Bar:** online.u2u.be/courses/25896/modules/11/24857
- Page Content:**
 - Section Title:** Web-specific Canonicalization
 - List:**
 - When applications make security decisions based on a name
 - When multiple representations of a resource are possible
 - Might allow security to be bypassed
 - You can secure the request by canonicalizing the name
 - Convert into a well-known format
 - Problem with web: many different representations for the same string
 - HtmlEncoding, UrlEncoding, ...
- Left Sidebar:** A navigation menu for the course, including sections like "Injection", "Cross-Site Scripting", "Cross-Site Request Forgery", "Session Hijacking", "Cookie Poisoning", "Cross-Site Scripting", "Run with Least Privilege", "Use Default MIME Types", "Web-specific Canonicalization", and "User Input Sanitization".
- Right Sidebar:** The U2U logo.
- Bottom Taskbar:** Shows the Windows Start button, a search bar, pinned icons for File Explorer, Task View, Control Panel, Internet Explorer, Google Chrome, and others, and system status icons for battery, signal, and date/time (6:39 PM, 12/29/2023).

The screenshot shows a web browser window with the title "U2U Online". The URL is "online.u2u.be/courses/25896/modules/11/24857". The page content is a slide titled "Hexadecimal Escape Codes". On the left, there's a sidebar with a navigation tree under "Injection" and a "Summary" section. On the right, there's a large "u2u" logo. The main content area contains two bullet points:

- Hexadecimal escape codes are a way to represent non-printable characters
 - Use %xx where xx is hexadecimal number representing that character
 - For example: space is %20
- Let's say you check in your code against hyphen (-)
 - Someone can easily bypass this check using %2D

At the bottom of the slide, there's a red link: <http://www.ascii.cl/htmlcodes.htm>.

The browser taskbar at the bottom shows the weather (25°C, Mostly cloudy), a search bar, and various pinned icons.

UTF-8 variable-width encoding

- UTF-8 can encode many different characters using multi-byte representation
 - ASCII 7-bit maps directly to UTF-8 8-bit representation with 0 leading bit.
- Other characters in range above 0x7F get encoded
 - For example: Pound Sterling is represented as 0xA3 (10100011 in binary)
 - Gets encoded as **11000010 10100011** (where **bold** are defined by the standard)
- Normally you should use the shortest encoding, but longer are allowed(*)
 - You could encode hyphen (-) which is 00101101 in binary as
 - **110 00000 10101101**

(*) allowed that is by badly implemented parsers!

The screenshot shows a web browser window with the following details:

- Title Bar:** U2U Online
- Address Bar:** online.u2u.be/courses/25896/modules/11/24857
- Left Sidebar (Course Navigation):**
 - Injection
 - Introduction
 - Visualizing the attack
 - JavaScript injection
 - Cross-site Scripting
 - Reflected XSS
 - DOM-based XSS
 - Run with Local storage
 - Use JavaScript integrity
 - Value from user input
 - Web-specific Considerations
 - User Input sanitization
 - Summary
- Content Area:**

HTML Escape Codes

 - HTML can escape characters using special characters
 - For example, < is represented by <
 - But there are other possibilities
 - For example, < can be represented by < (hex) and also < (decimal)
- Right Sidebar (u2u logo):** u2u
- Bottom Taskbar:**
 - 25°C Mostly cloudy
 - Search bar
 - Icons for File Explorer, Task View, Control Panel, Internet Explorer, Google Chrome, and others.
 - System icons: battery, signal, volume, and date/time (12/29/2023).

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Injection

- Injection
- SQL Injection
- XSS (Cross Site Scripting)
- Clickjacking
- Denial of Service
- Cross Site Scripting
- Denial of Service
- Run with Least Privilege
- Use Default/come Insecure
- Web Traffic Monitoring
- User Input Sanitization

Don't use parent paths in your application

■ Using parent paths (..) can lead to directory traversal

- Avoid ../images/Logo.jpg

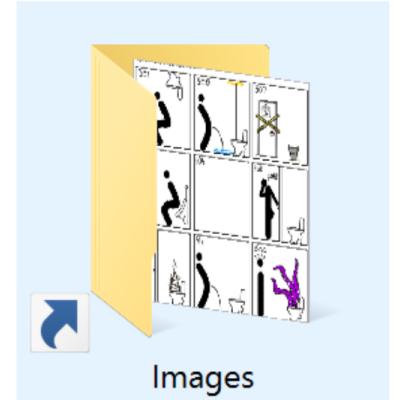
■ You can avoid parent paths by using symbolic links

- Symbolic links are like shortcuts to another file or folder

■ To create a symbolic link, use the **MKLINK** tool

- From a command prompt with Admin privileges
- For example: to create a link to the ..\images folder:

MKLINK /D Images ..\Images



25°C Mostly cloudy

Search

6:39 PM

The screenshot shows a web browser window with the following details:

- Title Bar:** U2U Online
- Address Bar:** online.u2u.be/courses/25896/modules/11/24857
- Page Content:**
 - Section Header:** Other web-based security topics
 - List:**
 - Don't trust the referer HTTP header
 - It can easily be spoofed
 - Don't put sensitive data in cookies or fields
 - Some web site puts the discount value in a hidden field
 - Then applies the discount from the hidden field during checkout
 - You can easily get a 50% discount at this store!

The sidebar on the left lists various security topics under the heading "Injection". The bottom of the screen shows a taskbar with icons for weather, search, and various applications.

The screenshot shows a web browser window with the following details:

- Title Bar:** u2u U2U Online
- Address Bar:** online.u2u.be/courses/25896/modules/11/24857
- Page Content:**
 - Section Title:** Simple user input remedies
 - List:**
 - Be hardcore about user input
 - Determine what input is valid and reject the rest (no exceptions!)
 - Approaches to white-listing input
 - Use conversions
 - Integers, Dates, GUIDs
 - A string that can be converted to an integer will not contain malicious input
 - Use regular expressions
 - E-mail, telephone numbers, names
 - Be careful not to be too restrictive!
 - List all known good values
 - Countries, departments, ...
- Sidebar:** Shows a navigation tree under 'Injection' with nodes like 'Reflection', 'SQL Injection', 'XSS (Cross Site Scripting)', etc.
- Bottom Status Bar:** Shows weather (25°C, Mostly cloudy), system icons (Search, File Explorer, Task View, Google Chrome), and system status (6:39 PM, ENG, battery level, 12/29/2023).

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/11/24857. The page title is "Hardcore checking with regular expressions". On the left, there's a sidebar with a navigation tree under "Injection" and a "Summary" section. The main content area contains a bulleted list and a code snippet. The bottom of the screen shows a taskbar with various icons and system status.

Hardcore checking with regular expressions

- For example, using Regular Expressions to check product name
 - Product name can only contain characters, digits and spaces
 - Product name has to be between 1 and 32 characters long

```
public class ProductChecker
{
    const string productNameRegEx = "^[A-Za-z0-9 ]{1,32}$";
    private readonly Regex regEx = new Regex(productNameRegEx);

    public bool CheckProductNameIsValid(string productName)
        => regEx.IsMatch(productName);
}
```

25°C Mostly cloudy Search 6:39 PM

u2u U2U Online

online.u2u.be/courses/25896/modules/11/24857

Home > Web Security Hardcore Techniques > Injection

Hardcore checking with regular expressions

u2u

- Checking the URI with MVC routing

```
[Route("product/{name:regex(^[A-Za-z0-9 ]+{1,32}$)})]
public IActionResult Product(string name)
```

- Verifying Form input

```
public class CommentViewModel
{
    [Required]
    [RegularExpression(@"^A-Za-z0-9\s.\.,\!\?\'\\""]+$")]
    public string Comment { get; set; }
}
```

25°C Mostly cloudy

Search

6:39 PM 12/29/2023

Not being strict is dangerous

- Some developers allow “safe” HTML constructs, e.g., or <table>
 - Do NOT allow this!
 - XSS exploit is still there

```
<img src=javascript:alert(document.domain)>
<link rel=stylesheet href="javascript:alert(document.domain)">
...

```

25°C Mostly cloudy 6:39 PM ENG 12/29/2023

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/11/24857. The page title is "Special case: Passwords". On the left, there's a sidebar with a navigation tree under "Injection" and a "Summary" section. The main content area contains a bulleted list:

- Try not to limit the allowed characters in a password, or its length
 - Always **hash** the password before storing/retrieving it
 - See **hashing** and **password policies**

The browser interface includes a top bar with tabs, a search bar, and various icons. The taskbar at the bottom shows the date and time (6:39 PM, 12/29/2023), weather (25°C, Mostly cloudy), and system status.

When input becomes output

- Restrict input to be absolutely safe (which you will output again)
- HtmlEncode everything
 - Easy in .NET
 - JavaScript requires library

Summary

- Always **parameterize untrusted data**
 - Never use string concatenation
- Always apply the **Principle of Least Privilege**
 - Does one need to be able to do that?
- Always **whitelist** untrusted data
 - Using conversions, regular expressions or lists
- Stored procedures offer better protection through parameterization
 - Unless you use string concatenation inside the SP!
- ORMs like Entity Framework automatically parameterize for security
- Never trust user input!