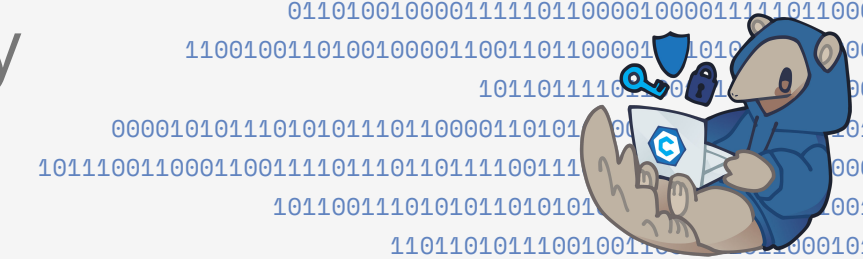
Cyber @ UCI

Build Secure Software:
Intro to Application Security



Attendance Form

For Event Attended:

“Intro to AppSec (WebJam)”

Today's Agenda

- ★ Why (Application) Security?
  - In Context of WebJam
- ★ Application Security Basics
- ★ Low-Hanging Fruits
  - Exposed Secrets / Credentials
  - SQL Injection
  - Cross-Site Scripting
  - Local File Inclusion
  - Insecure Direct Object References
- ★ Mini CTF

General Disclaimer

You should only practice the techniques demonstrated today through official websites that have dedicated exercises and training for it (or explicitly grant you permission to do so), or on your own code and environment. :)

If you do any vulnerability scanning and testing on open-source applications, you should report it to the maintainers in a responsible manner. :)

If you decide to search more about hacking, please do so ethically. :)

What is Application Security (AppSec)?

Subfield of security that focuses on protecting ***software applications and systems*** against both unintentional failures and malicious threats

- **Security-based processes** in the software development lifecycle
- **Code scanning automation** to prevent “low-hanging fruit” from showing up in production
- **Manual review** of code and **closed-box testing** (e.g., penetration testing)
- **Best practices to design** secure applications and systems

Why AppSec?

Chess.com discloses recent data breach via file transfer app

Chess.com has disclosed a data breach after threat actors gained unauthorized access to a third-party file transfer application used by the platform.

BILL TOULAS SEPTEMBER 04, 2025 01:51 PM 0

Hundreds of People With 'Top Secret' Clearance Exposed by House Democrats' Website

A database containing information on people who applied for jobs with Democrats in the US House of Representatives was left accessible on the open web.

LILY HAY NEWMAN AND MATT BURGESS

PhantomRaven Malware Found in 126 npm Packages Stealing GitHub Tokens From Devs

Oct 30, 2025 DevSecOps / Software Security

Cybersecurity researchers have uncovered yet another active software supply chain attack campaign targeting the npm registry with over 100 malicious packages that can steal...

Why AppSec?

Cisco's Wave of Actively Exploited Zero-Day Bugs Targets Firewalls, IOS

by **Alexander Culafi**

SEP 25, 2025 6 MIN READ

Google Pays $100,000 in Rewards for Two Chrome Vulnerabilities

The two bugs are high-severity type confusion and inappropriate implementation issues in the browser's V8 JavaScript engine.

APPLICATION SECURITY

Self-Replicating 'Shai-hulud' Worm Targets NPM Packages

SEP 16, 2025 | 4 MIN READ

by **Alexander Culafi**, Senior News Writer

In Context of WebJam...

Reality: Security probably isn't the focus of your web app.....
(unless you're making some kind of security-themed web application)

But ***considerations*** for it may put your project over the top!

- ★ Panel of professor judges → Bringing up security considerations in your presentation just adds another point of 'impression'
  - Be cautious about your understanding in case they have follow-up questions
  - Keep it simple – "I made sure SQL injection isn't a problem by doing ..."

- ★ If you decide to make your submission a cool side project, then it's good experience to keep in mind security
  - Resume builder and interview storytelling

AppSec-Related Tasks

- ★ Threat modeling – determining what security (and privacy) threats are viable and which ones should be prioritized
- ★ Auditing code for common vulnerabilities (or unique ones)
  - Manual code reviews
  - Code scanning with the help of tools
  - Automated scanning integrated in the CI/CD pipeline
- ★ Developing tooling and systems to support automation, system defenses, and mitigations
- ★ Creating security processes for developers to follow and adhere to (and to support company compliance)
- ★ Supporting dev and product teams with secure design

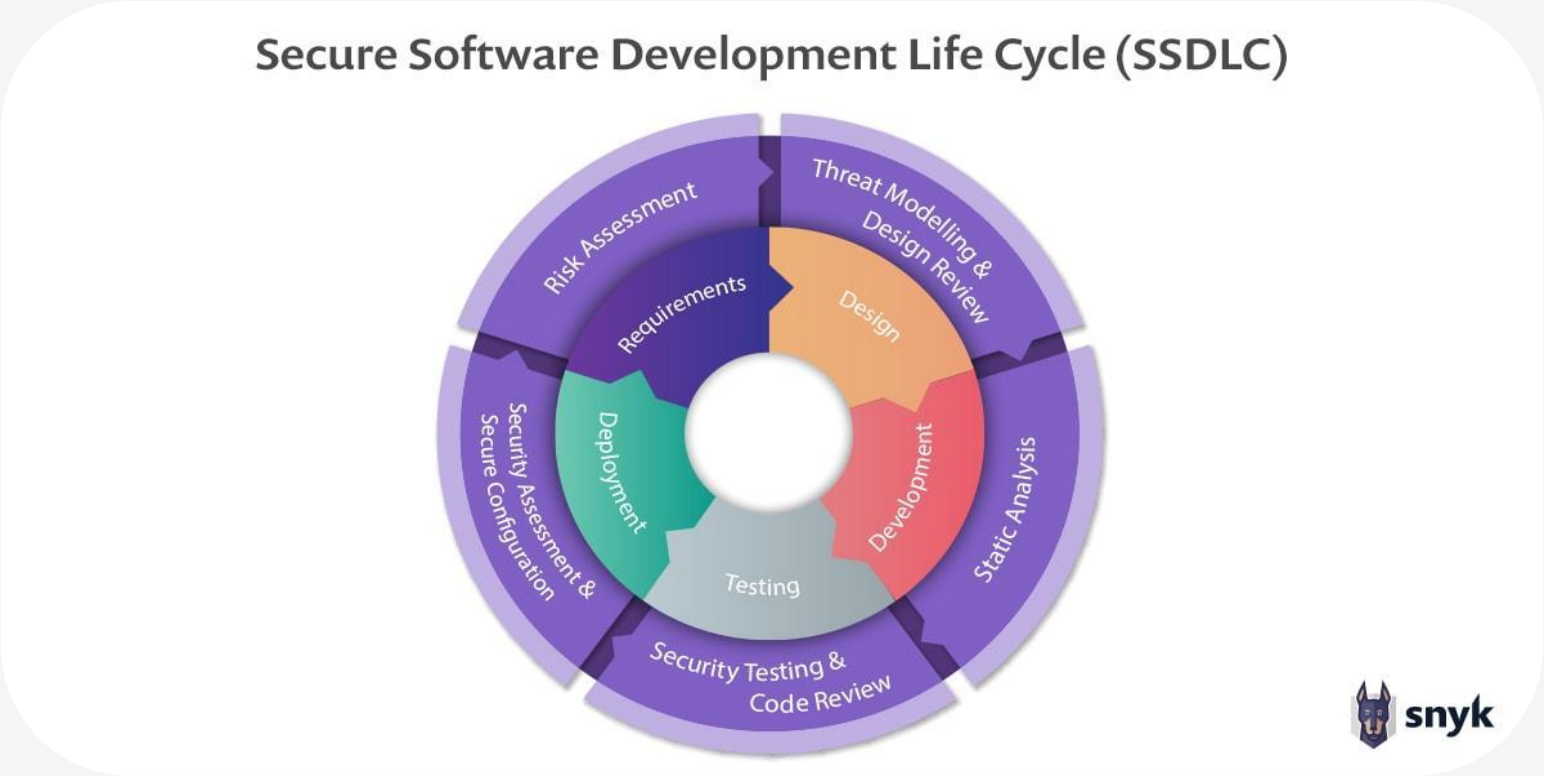
Relevant Features

- ★ **Authentication**: Are users logging in with **only** their own set of valid credentials?
- ★ **Authorization**: Are the users who are **supposed** to see certain things the only ones who can **actually** see them?
- ★ **Encryption**: Can sensitive data (at-rest and in-transit) be accessed **only** by those who have the correct key?
- ★ **Logging**: If there were any suspicious activities or malicious threats, do you have any **data** to support further investigations?
- ★ **Testing**: Is your code **actually** secure against what you **think** it's secure against?

Some Best Practices

- ★ **Authentication**: Implementing multi-factor auth, single sign-on; secure API auth standards (e.g., OAuth 2.0, JWT)
- ★ **Authorization**: Principle of least privilege, deny by default
- ★ **Encryption**: Don’t write your own encryption! Use approved algorithms (e.g., AES-256, RSA 2048, SHA-256)
- ★ **Logging**: Descriptive error messages, tracking third-party dependencies (e.g., Software Bill of Materials (SBOMs))
- ★ **Testing**: Automate your continuous testing (e.g., CI/CD), utilize static & dynamic analysis

SDLC → Secure SDLC



OWASP Top 10

Top 10 list of “most critical security risks to web application”; last published in 2021

#1: Broken Access Control

#2: Cryptographic Failures

#3: Injection

#4: Insecure Design

#5: Security Misconfiguration

#6: Vulnerable and Outdated Components

#7: Identification and Authentication Failures

#8: Software and Data Integrity Failures

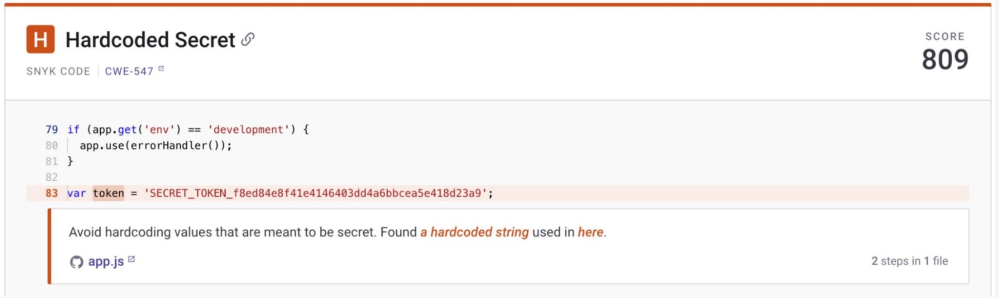
#9: Security Logging and Monitoring Failures

#10: Server-Side Request Forgery (SSRF)

Exposed Secrets / Credentials

Especially if your code repo is public, you have to be careful about exposing your:

- API keys
- Passwords
- SSH keys
- Access tokens
- Database connection strings
- SaaS account IDs



Even then, secrets can be stolen by various means (i.e., other upcoming methods..)

Also be cautious about leaving secrets in comments!

SQL Injection

Overall Goal: Leveraging unsanitized SQL queries to your advantage

Target: Login to the user '**stengo**' without knowing their password

Query:

```
"SELECT * FROM users WHERE username = '{username}' AND password = '{password}'"
```

Thoughts on what you could do here?

Hints:
Add something in the username input that comments the rest of the check out...
OR add another component to the query that causes it to always be true...

SQL Injection

Query:

“SELECT * FROM users WHERE username = ‘{username}’ AND password = ‘{password}’“

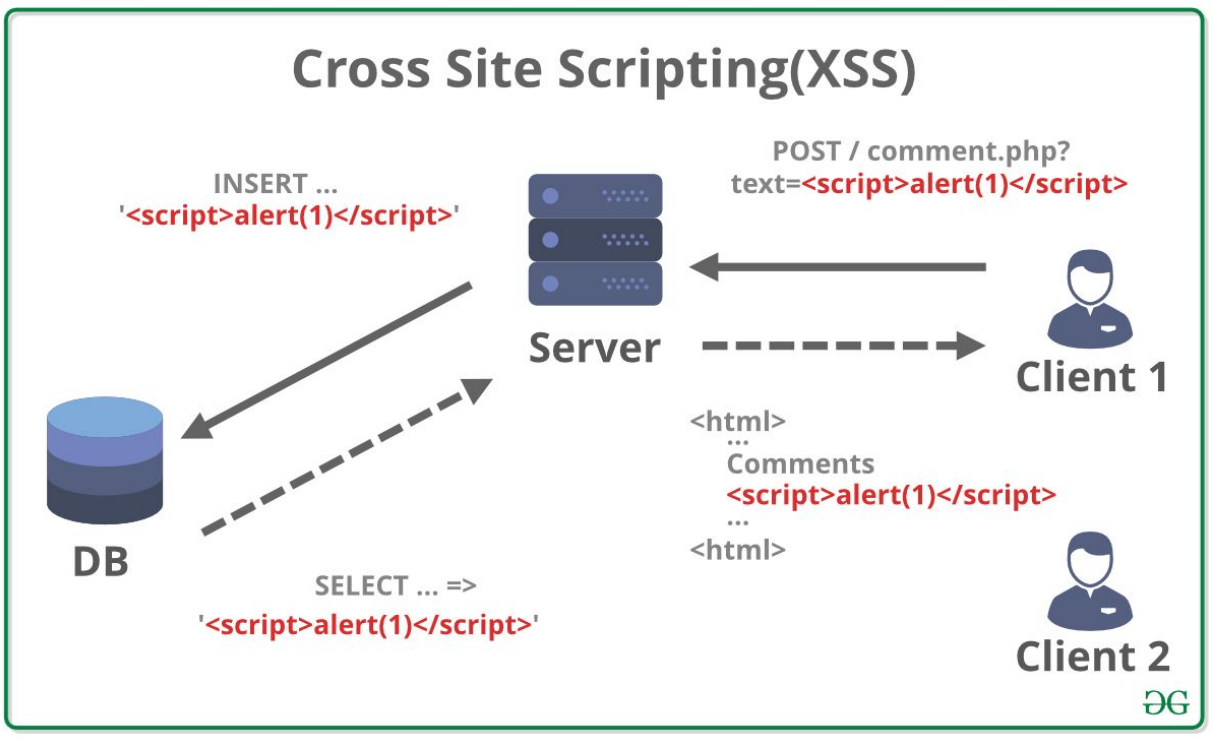
Modified Query (with input):

“ SELECT * FROM users WHERE **username = ‘stengo’--’** AND password = ‘{password}’ “

Effective Query Sent:

“ SELECT * FROM users WHERE username = ‘stengo’ ”

Cross-Site Scripting



Cross-Site Scripting

Three main kinds of XSS:

1. **Reflected** = From clicking on a malicious link that leads to the application to "reflect" the script in the site's response, causing victim's browser to execute it

2. **Stored** = Malicious script is stored on the server/application itself, so that whenever someone then renders that script (e.g., forum post, comment field), it is executed on the victim's browser

3. **Document Object Model (DOM)-based** = Exploiting the DOM "environment" of the victim's browser (e.g., web app's JavaScript writes to the DOM w/o proper sanitization)

Cross-Site Scripting

Focusing on **Stored XSS**...

Before Payload

```
<!doctype html><meta
charset="utf-8"><title>Before</title>

<div id=out></div>

<p> Hi! This is my first forum post. </p>

...

<p> Hello! This is my second post. </p>

...

<p>[Where the next post would be rendered] </p>
```

After Payload

```
<!doctype html><meta
charset="utf-8"><title>Before</title>

<div id=out></div>

<p> Hi! This is my first forum post. </p>

...

<p> Hello! This is my second post. </p>

...

<p> <script> alert("1") </script> </p>
```

Local File Inclusion

Scenario: Imagine if your website directly accesses the local file system for a specific reason (e.g., to load images, render file text) → image repository

URL: `https://catimages.com/?img=calico_cat1.png`

(Fictional) code behind the scenes:

```
filename = request.args.get('img', '')

with open(filename, 'r') as f:
    content = f.display()
```

Local File Inclusion

Assume: The web server is hosted on a Linux server, so you can use the Linux file path formatting to your advantage.....

Modified URL: `https://catimages.com/?img=../../../../../etc/passwd`

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
```

Insecure Direct Object References (IDOR)

Picture this: After you login to your bank account, you get the following URL -

`https://mellsmargo.com/account?user_id=2365832`

What component of the URL would be **different for each user** of the bank's website?

`https://mellsmargo.com/account?user_id=1`
`https://mellsmargo.com/account?user_id=2365831`
`...`

IDORs


ZERO THREAT
Types of IDOR Attacks
Path Traversal
URL Tampering
Modifying Cookie/JSON ID
Body Manipulation
zerothreat.ai

AppSec CTF!

1. Go to ctfd.cyberuci.com, and create an account
2. Go to the “Challenges” tab
3. Go to vuln-site.cyberuci.com
4. Find the flags!

TL;DR...

- A good chunk of these “low-hanging fruits” can be fixed by adding some sort of input sanitization or implementing additional authorization checks to resources

- The challenges

Tools of Interest

Note: A lot of these tools are free for your own research and personal projects, but have an enterprise option when they are used by for-profit companies!

- ★ Semgrep: https://github.com/semgrep/semgrep
- ★ Aikido: https://github.com/AikidoSec
  - Software Supply Chain: https://github.com/AikidoSec/safe-chain
  - Software Composition Analysis: https://www.aikido.dev/scanners/open-source-dependency-scanning-sca
- ★ ZAP: https://www.zaproxy.org/
- ★ Trivy: https://trivy.dev/latest/
- ★ FuzzForge: https://github.com/FuzzingLabs/fuzzforge_ai
  - “AI-Powered Workflow Automation” - experimental!

Additional Resources

- ★ Open Worldwide Application Security Project:
  - https://owasp.org/Top10/
  - https://owasp.org/www-project-api-security/
- ★ API Security.io:
  - https://apisecurity.io/encyclopedia/content/owasp/owasp-api-security-top-10.htm
- ★ Snyk:
  - https://snyk.io/learn/application-security/
  - https://snyk.io/blog/10-react-security-best-practices/
- ★ NodeJS:
  - https://nodejs.org/en/docs/guides/security/
- ★ OWASP Cheat Sheet Series:
  - https://cheatsheetseries.owasp.org/cheatsheets/Forgot_Password_Cheat_Sheet.html
- ★ Gitlab:
  - https://docs.gitlab.com/ee/user/application_security/dependency_scanning/

Join Cyber @ UCI

For more trainings on appsec and general security, come to our meetings!

- Join our Discord: discord.cyberuci.com
- Follow our Instagram: @ cyberuci
- Our Fall ‘25 meetings are on Wednesdays from 6:30pm - 7:30pm
  - Intro to Ethical Hacking next week!
  - Guest speakers from Splunk/Cisco in 2 weeks!
- Recruitment for various operation and competition teams throughout the year
  - Join our Discord for the latest announcements!