Welcome to OWASP

Sioux Falls Chapter
Announcements

- Infragard meeting this Thursday, Jan. 18
  - Food at 1:00 - meeting at 1:30
  - University Center Room 253
  - Open to the public
EXPLOITS OF A MOM

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 BOBJOY 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.

https://xkcd.com/

About OWASP

• Founded in December, 2001
• Open professional organization
• Focused on educating others about secure application development
• Numerous major projects being sponsored by the organization
• Projects need volunteers!
What OWASP Is

• 501c3 Non-Profit Professional Organization
• An international organization
• Free and open to anyone interested in learning about application security
• Maintainer of the OWASP Top 10 Project
• Maintainer of numerous open tools
• The sum of its volunteer members
What OWASP Is Not

• Affiliated with any technology company *
• A hacker club *
• Exclusionary
• An opportunity to advertise
• Solely beholden to application security
Why Sioux Falls?

- Growing community of IT pros
- Training options are limited
- Collaboration is good
- South Dakota has great resources
  - DSU, SDSM&T, SDSU, USD, et al.
  - Black Hills InfoSec/Wild West Hack Fest
- Why not Sioux Falls?
- Why not South Dakota?
Who Should Attend OWASP?

- Operational IT professionals of all stripes
- Executives and management
- Novices and students
- Hobbyists
- Anyone looking to learn
- CISSPs looking to earn more CPE credits
Membership

- Don’t have to be a member to attend
- Don’t have to be a member to present
- Don’t have to be a member to lead
- $50 a year if you decide to join
- 40% of dues can go to the local chapter
- Membership dues help our chapter operate
- https://www.owasp.org/index.php/Membership
Sioux Falls OWASP

• Leadership
  • Paul Kern - chapter leader
  • Joey Henkel - chapter co-leader
  • Scott Francis
  • Amanda Marczak
  • You?

• Chapter leaders will be voted on every two years.
Meeting Ideas

• We are always looking for meeting ideas
  • Speakers
  • Topics
  • Format suggestions
  • Venue suggestions
  • Time and date
• We aim to have quarterly meetings
• paul.kern@owasp.org
• joey.henkel@owasp.org
More Information

- [https://www.owasp.org/index.php/Main_Page](https://www.owasp.org/index.php/Main_Page)
- [https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project](https://www.owasp.org/index.php/About_The_Open_Web_Application_Security_Project)
- [https://www.owasp.org/index.php/Sioux_Falls](https://www.owasp.org/index.php/Sioux_Falls)
- [https://twitter.com/siouxfallsowasp](https://twitter.com/siouxfallsowasp)
- [https://www.facebook.com/OWASP-Sioux-Falls-1633373690053738/](https://www.facebook.com/OWASP-Sioux-Falls-1633373690053738/)
What is the OWASP Web Top 10?

• Original intent was to raise awareness.
• Has become the de facto application security standard.
• The latest version of the list was just released at the end of 2017.
• Previous version was released in 2013.
• Lots has changed in four years.
WHAT HAS CHANGED SINCE 2013?
What has changed since 2013?

- JavaScript is now the king.
- Angular, Bootstrap, React, etc. on the client-side.
- Node.js handling the server-side
- Monolithic applications now being replaced with microservices.
- Advantages and drawbacks abound.
What has changed since 2013?

<table>
<thead>
<tr>
<th>OWASP Top 10 - 2013</th>
<th>OWASP Top 10 - 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 - Injection</td>
<td>A1: 2017 - Injection</td>
</tr>
<tr>
<td>A2 - Broken Authentication and Session Management</td>
<td>A2: 2017 - Broken Authentication</td>
</tr>
<tr>
<td>A3 - Cross-Site Scripting (XSS)</td>
<td>A3: 2017 - Sensitive Data Exposure</td>
</tr>
<tr>
<td>A4 - Insecure Direct Object References [Merged + A7]</td>
<td>A4: 2017 - XML External Entities (XXE) [NEW]</td>
</tr>
<tr>
<td>A5 - Security Misconfiguration</td>
<td>A5: 2017 - Broken Access Control [Merged]</td>
</tr>
<tr>
<td>A6 - Sensitive Data Exposure</td>
<td>A6: 2017 - Security Misconfiguration</td>
</tr>
<tr>
<td>A8 - Cross-Site Request Forgery (CSRF)</td>
<td>A8: 2017 - Insecure Deserialization [NEW, Community]</td>
</tr>
<tr>
<td>A9 - Using Components with Known Vulnerabilities</td>
<td>A9: 2017 - Using Components with Known Vulnerabilities</td>
</tr>
<tr>
<td>A10 - Unvalidated Redirects and Forwards</td>
<td>A10: 2017 - Insufficient Logging &amp; Monitoring [NEW, Community]</td>
</tr>
</tbody>
</table>
A4:2017 - XML External Entities (XXE)

<table>
<thead>
<tr>
<th>OWASP Top 10 - 2013</th>
<th>OWASP Top 10 - 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2 – Broken Authentication and Session Management</td>
<td>A2:2017-Broken Authentication</td>
</tr>
<tr>
<td>A3 – Cross-Site Scripting (XSS)</td>
<td>A3:2017-Sensitive Data Exposure</td>
</tr>
<tr>
<td>A4 – Insecure Direct Object References [Merged+A7]</td>
<td><strong>A4:2017-XML External Entities (XXE) [NEW]</strong></td>
</tr>
<tr>
<td>A6 – Sensitive Data Exposure</td>
<td>A6:2017-Security Misconfiguration</td>
</tr>
<tr>
<td>A8 – Cross-Site Request Forgery (CSRF)</td>
<td>A8:2017-Insecure Deserialization [NEW, Community]</td>
</tr>
<tr>
<td>A9 – Using Components with Known Vulnerabilities</td>
<td>A9:2017-Using Components with Known Vulnerabilities</td>
</tr>
<tr>
<td>A10 – Unvalidated Redirects and Forwards</td>
<td>A10:2017-Insufficient Logging &amp; Monitoring [NEW, Community]</td>
</tr>
</tbody>
</table>
A4:2017 - XML External Entities (XXE)

Attack Vector

• Exploit vulnerable XML processors
  • many older XML processors allow access to external entities (aka URIs)
  • URIs are evaluated during XML processing
  • exploit XML processor using hostile content
  • extract data from system
  • remote code execution on server
Example
The attacker attempts to extract data from the server:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
<!ELEMENT foo ANY >
<!ENTITY xxe SYSTEM "file:///etc/passwd" >]
<foo>&xxe;</foo>
```
A4:2017 - Who is at risk?

- Application accepts and processes direct XML or XML uploads
- XML processors or SOAP-based web services with document type definitions (DTD) enabled.
- Application uses SAML for SSO
- SOAP pre version 1.2
A4:2017 - What can be done?

• Use less complex formats like JSON
• Disable XML external entity and DTD processing
• Positive server-side input validation, filtering, sanitation
• W3C XML validation (XSD)
• https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet
A8:2017 - Insecure Deserialization

<table>
<thead>
<tr>
<th>OWASP Top 10 - 2013</th>
<th>OWASP Top 10 - 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 - Injection</td>
<td>A1:2017-Injection</td>
</tr>
<tr>
<td>A2 - Broken Authentication and Session Management</td>
<td>A2:2017-Broken Authentication</td>
</tr>
<tr>
<td>A3 - Cross-Site Scripting (XSS)</td>
<td>A3:2017-Sensitive Data Exposure</td>
</tr>
<tr>
<td>A4 - Insecure Direct Object References [Merged+A7]</td>
<td>A4:2017-XML External Entities (XXE) [NEW]</td>
</tr>
<tr>
<td>A5 - Security Misconfiguration</td>
<td>A5:2017-Broken Access Control [Merged]</td>
</tr>
<tr>
<td>A6 - Sensitive Data Exposure</td>
<td>A6:2017-Security Misconfiguration</td>
</tr>
<tr>
<td>A8 - Cross-Site Request Forgery (CSRF)</td>
<td>A8:2017-Insecure Deserialization [NEW, Community]</td>
</tr>
<tr>
<td>A9 - Using Components with Known Vulnerabilities</td>
<td>A9:2017-Using Components with Known Vulnerabilities</td>
</tr>
<tr>
<td>A10 - Unvalidated Redirects and Forwards</td>
<td>A10:2017-Insufficient Logging &amp; Monitoring [NEW, Community]</td>
</tr>
</tbody>
</table>
• Serialization - process of converting an object into a format that can be persisted to disk or sent across a network.
  • Can be binary
  • Structured Data (JSON and XML)
• Deserialization - the opposite. Converting data from a file, stream or network to an object to processed on the system.
• Affects most programming languages (JAVA)
A8:2017 - Insecure Deserialization

Attack Vector

• Occurs when untrusted data is used to abuse the logic of an application to cause DoS or cause remote code execution
• Somewhat difficult to pull off
• Off-the-shelf exploits almost always require tweaks to the underlying code
• Advanced Skill-set
A8:2017 - Insecure Deserialization

Example

A PHP forum uses PHP object serialization to save a "super" cookie, containing the user's user ID, role, password hash, and other state:

```
a:4:{i:0;i:132;i:1;s:7:"Mallory";i:2;s:4:"user";i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```

An attacker is able to send tampered, serialized cookie data to give herself admin privileges when it is deserialized:

```
a:4:{i:0;i:1;i:1;s:5:"Alice";i:2;s:5:"admin";i:3;s:32:"b6a8b3bea87fe0e05022f8f3c88bc960";}
```
A8:2017 - Who is at risk?

- Applications or APIs that could potentially deserialize tampered objects
- Pretty vague, huh?
- There are multiple ways that it can be accomplished.
- Varies by language, application structure, data flow, etc.
A8:2017 - What can be done?

- If possible, don’t accept serialized object from untrusted sources
- If possible, restrict to only primitive data types
- Integrity checks/digital signatures
- Isolate and run deserialization code in low privilege environments
- Logging and monitoring

https://www.owasp.org/index.php/Deserialization_Cheat_Sheet
<table>
<thead>
<tr>
<th>OWASP Top 10 - 2013</th>
<th>OWASP Top 10 - 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2 – Broken Authentication and Session Management</td>
<td>A2:2017-Broken Authentication</td>
</tr>
<tr>
<td>A3 – Cross-Site Scripting (XSS)</td>
<td>A3:2017-Sensitive Data Exposure</td>
</tr>
<tr>
<td>A6 – Sensitive Data Exposure</td>
<td>A6:2017-Security Misconfiguration</td>
</tr>
<tr>
<td>A8 – Cross-Site Request Forgery (CSRF)</td>
<td>A8:2017-Insecure Deserialization [NEW, Community]</td>
</tr>
<tr>
<td>A9 – Using Components with Known Vulnerabilities</td>
<td>A9:2017-Using Components with Known Vulnerabilities</td>
</tr>
<tr>
<td>A10 – Unvalidated Redirects and Forwards</td>
<td>A10:2017-Insufficient Logging &amp; Monitoring [NEW, Comm]</td>
</tr>
</tbody>
</table>
A10:2017 - Insufficient Logging & Monitoring

Attack Vector

- Nearly all major incidents have this item as their foundation
- Response time is critical
- Attackers rely on slow response times and poor logging to successfully pull off their attacks.
A10:2017 - Who is at risk?

- Are you logging all authorization attempts, failed or otherwise?
- Are you logging warnings and errors?
- Are logs stored local to the application?
- Who is monitoring the logs and how often?
- What are alerting thresholds?
- Is alerting real-time?
A10:2017 - What can be done?

- Standardize application logs and make sure they can be consumed by a SIEM
- Strong log monitoring processes
- Log all authentication attempts, warnings, input validation errors, etc.
- Identify and log high value transactions
- Develop an incident response plan
- https://www.owasp.org/index.php/Logging_Cheat_Sheet
A5:2017 - Broken Access Control

<table>
<thead>
<tr>
<th>OWASP Top 10 - 2013</th>
<th>OWASP Top 10 - 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1 - Injection</td>
<td>A1:2017-Injection</td>
</tr>
<tr>
<td>A2 - Broken Authentication and Session Management</td>
<td>A2:2017-Broken Authentication</td>
</tr>
<tr>
<td>A3 - Cross-Site Scripting (XSS)</td>
<td>A3:2017-Sensitive Data Exposure</td>
</tr>
<tr>
<td>A4 - Insecure Direct Object References [Merged+A7]</td>
<td>A4:2017-XML External Entities (XXE) [NEW]</td>
</tr>
<tr>
<td>A5 - Security Misconfiguration</td>
<td>A5:2017-Broken Access Control [Merged]</td>
</tr>
<tr>
<td>A6 - Sensitive Data Exposure</td>
<td>A6:2017-Security Misconfiguration</td>
</tr>
<tr>
<td>A8 - Cross-Site Request Forgery (CSRF)</td>
<td>A8:2017-Insecure Deserialization [NEW, Community]</td>
</tr>
<tr>
<td>A9 - Using Components with Known Vulnerabilities</td>
<td>A9:2017-Using Components with Known Vulnerabilities</td>
</tr>
<tr>
<td>A10 - Unvalidated Redirects and Forwards</td>
<td>A10:2017-Insufficient Logging &amp; Monitoring [NEW, Community]</td>
</tr>
</tbody>
</table>

[OWASP Open Web Application Security Project]
A5:2017 - Broken Access Control

Attack Vector

• Core skill for most attackers
• This topic is very broad and not specific
• Multiple scenarios that can fall in this category
• SAST and DAST can test for absence of security controls, but can’t verify functionality.
• Manual testing required
A5:2017 - Broken Access Control

Example
An application uses unverified data in a SQL call that is accessing account information:
```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery();
```

An attacker simply modifies the 'acct' parameter in the browser to send whatever account number they want. If not properly verified, the attacker can access any user's account.

```
http://example.com/app/accountInfo?acct=notmyacct
```
A5:2017 - Who is at risk?

- Can access control be bypassed by modifying the URL, HTML, or application state?
- Can I alter a DB primary key and access another user’s data?
- Can I accomplish privilege escalation?
- Can I force browse to authenticated pages an unauthenticated user?
- Can I force browse to admin as a non admin?
A5:2017 - What can be done?

• Non public resources? Deny by default.
• Implement access control mechanisms once and reuse them.
• Enforce user record ownership controls
• Disable directory listing/clean up root

- [https://www.owasp.org/index.php/Access_Control_Cheat_Sheet](https://www.owasp.org/index.php/Access_Control_Cheat_Sheet)
Gone, But Not Forgotten

<table>
<thead>
<tr>
<th>OWASP Top 10 - 2013</th>
<th>OWASP Top 10 - 2017</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A1</strong> - Injection</td>
<td><strong>A1:2017</strong> - Injection</td>
</tr>
<tr>
<td><strong>A2</strong> - Broken Authentication and Session Management</td>
<td><strong>A2:2017</strong> - Broken Authentication</td>
</tr>
<tr>
<td><strong>A3</strong> - Cross-Site Scripting (XSS)</td>
<td><strong>A3:2017</strong> - Sensitive Data Exposure</td>
</tr>
<tr>
<td><strong>A4</strong> - Insecure Direct Object References [Merged+A7]</td>
<td><strong>A4:2017</strong> - XML External Entities (XXE) [NEW]</td>
</tr>
<tr>
<td><strong>A5</strong> - Security Misconfiguration</td>
<td><strong>A5:2017</strong> - Broken Access Control [Merged]</td>
</tr>
<tr>
<td><strong>A6</strong> - Sensitive Data Exposure</td>
<td><strong>A6:2017</strong> - Security Misconfiguration</td>
</tr>
<tr>
<td><strong>A8</strong> - Cross-Site Request Forgery (CSRF)</td>
<td><strong>A8:2017</strong> - Insecure Deserialization [NEW, Community]</td>
</tr>
<tr>
<td><strong>A9</strong> - Using Components with Known Vulnerabilities</td>
<td><strong>A9:2017</strong> - Using Components with Known Vulnerabilities</td>
</tr>
<tr>
<td><strong>A10</strong> - Unvalidated Redirects and Forwards</td>
<td><strong>A10:2017</strong> - Insufficient Logging &amp; Monitoring [NEW, Comm.]</td>
</tr>
</tbody>
</table>
Gone But Not Forgotten

• A8 Cross Site Request Forgery
  • Less than 5% of the data set today supports CSRF. Places it at about number 13.

• A10 Unvalidated redirects and forwards.
  • Less than 1% of the data set supports this issue today. Places it around number 25

• Do we ignore these if found?
OWASP 2017 Top 10 RC2

QUESTIONS?

paul.kern@owasp.org