APEX Security Checklist

Scott Spendolini
Vice President, APEX+ Practice

VISCOSITY
NORTH AMERICA

Welcome

About Me

scott@sumnertech.com
@sspendol

Agenda

- Overview
- Top Ten Threats
- Summary

Viscosity @ RMOUG

- **Tuesday, February 19**
- *9:00 AM – 12:00 PM | Standley 1 | Charles Kim, Jerry Ward, Scott Spendolini | APEX for the DBA, Pre-Conference Workshop*
- *1:00 PM – 2:00 PM | Meadowbrook 1 | Scott Spendolini | At Your Service: Web Services & APEX*
- 3:45 PM – 4:45 PM | Standley 1 | Nitin Vengulekar | Oracle Autonomous Data Warehouse Cloud: Testing, Experiences, Results
- **Wednesday, February 20**
- 8:30 AM – 9:30 AM | Windsor | Nitin Vengurlekar | Oracle cloud for EBS/Exadata Cloud Service: From Planning to Provisioning
- 8:30 AM – 9:30 AM | Standley 1 | Rich Niemiec | The Oracle 18c Best New Features & a Few 12cr2 Tips
- 1:30 PM – 2:30 PM | Standley 1 | Charles Kim | Get Ready for Brain Overload with Oracle Database 12.2 & 18c Features
- ***1:30 PM – 2:00 PM | Meadowbrook 2 | Scott Spendolini | APEX Security Checklist***
- 4:15 PM – 5:15 PM | Cotton Creek 1 | Charles Kim | Bulletproof Your Data Guard with Best Practices
- 6:30 PM – 9:30 PM | Westin Westminster | Happy Hour at Kachina Southwest Grill
- **Thursday, February 21**
- 11:15 AM – 12:15 PM | Standley 1 | Rich Niemiec | Innovation, the Oracle Cloud, Big Data, & The Internet of Things

Overview

OWASP Top 10

- **Open Web Application Security Project (OWASP)**
  - https://www.owasp.org/index.php/Main_Page
  - **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 - Injection**
- **A2:2017 - Broken Authentication**
- **A3:2017 - Sensitive Data Exposure**
- A4:2017 - XML External Entities (XXE)
  - Can be largely ignored in most cases, unless you're uploading and processing XML files
- **A5:2017 - Broken Access Control**
- **A6:2017 - Security Misconfiguration**
- **A7:2017 - Cross-Site Scripting (XSS)**
- **A8:2017 - Insecure Deserialization**
- **A9:2017 - Using Components with Known Vulnerabilities**
- **A10:2017 - Insufficient Logging & Monitoring**

Top 10 Threats

A1:2017 Injection

Tampering

- **Every web application** is subject to tampering
- Malicious users may try to exploit weaknesses in your APEX application using a number of different techniques
  - Page Attributes
  - URL Tampering
  - SQL Injection
  - Cross Site Scripting
- Fortunately, steps can be taken to prevent these types of attacks

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

URL Tampering

URL Tampering

- Consider this scenario:
  - An authenticated, legitimate yet malicious and/or curious user logs on to your application
  - He notices that when he hovers the mouse over the Edit link on Page 2, the end of the URL looks something like this:

```
...:P2_EMPNO:10
```

  - Curious, he manually changes the URL in his browser to read:

```
...:P2_EMPNO:20
```

  - And he is now viewing Department 20, which he should not be able to

DEMO: Change value in the URL to 7499

URL Tampering

- This is called **URL Tampering**
  - One of the most dangerous forms of attacks, as:
    - No programming is required
    - Anyone can do it
    - Developers do not always protect against it
    - Results can be disastrous!
- Essentially, a clever, malicious user can **alter the value of their session state** by passing item & value pairs through the URL
  - Unless precautions are taken

Session State Protection

- **Session State Protection** is a feature in APEX that combats URL Tampering
  - Generates an additional Checksum and passes that as part of the URL
  - If the Checksum is absent or altered, the page will not render, and thus the values will not be set
  - Must be enabled at the Application Level for it to work
    - Shared Components > Security > Session State Protection

DEMO: Enable SSP and note the difference

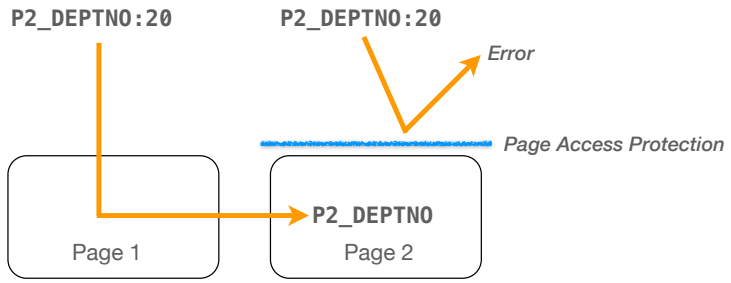
Page Access Protection

- Once Session State Protection is enabled, **Page Access Protection** should then be enabled for all pages in your application
- Four options for Page Access Protection
  - **Unrestricted**
    - Default and Least Secure
  - **Arguments Must Have Checksum**
  - **No Arguments Allowed**
  - **No URL Access**

DEMO: Enable PAP at each level

Page Access Protection Warning

- Page Access Protection **is not always enough**
  - A malicious user can set an item on Page 2 by passing values to that item via Page 1 and then changing the URL to view Page 2



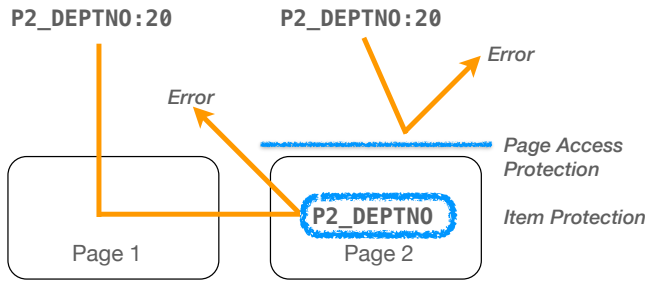
DEMO: Set page 2 item from page 1

Item Protection

- Item Protection will restrict how an item's value can be set
- Five Options:
  - **Unrestricted**
    - Default and Least Secure
  - **Checksum Required - Application**
  - **Checksum Required - User**
  - **Checksum Required - Session**
  - **Restricted - may not be set from a browser**

Item Protection

- With Item Protection enabled, an additional checksum needs to be present or an item's value cannot be changed via the URL



DEMO: Set page 2 item from page 1

Hidden Item Tampering

Hidden Items

- Hidden items **do not display** when an HTML page is rendered
  - But, they can contain a value that is sent back to the server when the page is POSTed
- While this value is not displayed, that doesn't mean that it can't be easily edited by a malicious user

Hidden Items

- Example of a Hidden Item in APEX:

```
<input type="hidden" id="P1_ID" name="p_t01" value="123" />
```

- Without item-level protection enabled, a user could - using embedded browser tools - change the value of that item to 456 and submit the page
  - Depending on what this item is used for, that could be disastrous

 DEMO: Alter a hidden item and submit the page

Hidden Item Protection

- Fortunately, this condition is **easy to mitigate with Item Level protection**
  - Option for Hidden Items that when enabled, will produce a checksum alongside the hidden item
  - When the page is submitted, if either the checksum or item value is altered, APEX will not process the page

 DEMO: Set Value Protected to Yes and retry

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

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

Bind Variables

- Be careful when using
  - `DBMS_SQL`
  - `EXECUTE IMMEDIATE`
- Always use **Bind Variables** where ever possible
- When you are forced to use **&ITEM.** notation
  - Be aware where the data in those items is coming from
    - APEX application, other web application, web service, etc.
  - When in doubt, escape it before rendering

Bind Variables

- Beware of **Bind Variables in Dynamic SQL**
  - The **use of bind variables** alone **does not eliminate** the potential for **SQL Injection**
  - Consider this example:

```
l_sql := 'SELECT * FROM emp
  WHERE empno =' || :P1_EMPNO;
RETURN l_sql;
```

  - It's no better than this:

```
SELECT * FROM emp
  WHERE empno = &P1_EMPNO.
```


Bind Variables

- Thus, in **Dynamic SQL**, be sure to **embed the bind variables in the string**, so that when the query executes, they appear as bind variables, not evaluated values
  - Correct usage in Dynamic SQL:

```
l_sql := 'SELECT * FROM emp
  WHERE empno = :P1_EMPNO';
RETURN l_sql;
```

DBMS_ASSERT

- Use **DBMS_ASSERT**
  - Introduced in 10g, **DBMS_ASSERT** is used to sanitize user input
- Main goal is to **guard against SQL injection attacks** by either sanitizing or validating user input before it's executed
  - Done by calling individual functions before passing user input to a string that will be executed
- If values have been tampered with or are not legitimate, **DBMS_ASSERT** will fail
  - And the SQL should not be executed

DBMS_ASSERT

- DBMS_ASSERT Functions:
  - **ENQUOTE_LITERAL**
  - **ENQUOTE_NAME**
  - **NOOP**
  - **QUALIFIED_SQL_NAME**
  - **SCHEMA_NAME**
  - **SIMPLE_SQL_NAME**
  - **SQL_OBJECT_NAME**
- See https://www.owasp.org/index.php/PL/SQL_Security_Cheat_Sheet for more details

Example: DBMS_ASSERT

```
PROCEDURE get_empno
  (
  p_code IN VARCHAR2
  )
IS
  l_sql     VARCHAR2(32767);
  c_cursor  SYS_REFCURSOR;
  l_buffer  VARCHAR2(32767);
BEGIN

  l_sql := 'SELECT empno FROM emp WHERE ename = ''' || p_code || '''';

  OPEN c_cursor FOR l_sql;
  LOOP
    FETCH c_cursor
    INTO  l_buffer;
    EXIT WHEN c_cursor%NOTFOUND;

    DBMS_OUTPUT.put_line(l_buffer);
  END LOOP;
END;
/
```


Example: DBMS_ASSERT

- Input

```
BEGIN
get_empno(p_code => 'KING');
END;
/
```

- Output

```
7839
```

Example: DBMS_ASSERT

- Input

```
BEGIN
get_empno(p_code => 'KING'' OR ''1''=''1');
END;
/
```

- Output

```
7369                    7499
7521                    7566
7654                    7698
7782                    7788
7839                    7844
7876                    7900
7902                    7934
```


Example: DBMS_ASSERT

- Input

```
BEGIN
get_empno(p_code => 'KING'' UNION SELECT deptno FROM
  dept WHERE ''1''=''1');
END;
/
```

- Output

```
10
20
30
40
7839
```


Example: DBMS_ASSERT

```
PROCEDURE get_empno
  (
  p_code IN VARCHAR2
  )
IS
  l_sql     VARCHAR2(32767);
  c_cursor  SYS_REFCURSOR;
  l_buffer  VARCHAR2(32767);
BEGIN

  l_sql := 'SELECT empno FROM emp WHERE ename = ' ||
    SYS.DBMS_ASSERT.ENQUOTE_LITERAL(p_code);

  OPEN c_cursor FOR l_sql;
  LOOP
    FETCH c_cursor
    INTO  l_buffer;
    EXIT WHEN c_cursor%NOTFOUND;

    DBMS_OUTPUT.put_line(l_buffer);
  END LOOP;
END;
/
```


Checklist

- ✓ Enable **Session State Protection**
- ✓ Enable **Page Access Protection** for ALL pages
- ✓ Ensure that all **Hidden Items** are **Protected**
- ✓ Use **bind variables** in any SQL & PL/SQL to avoid SQL Injection
- ✓ Use **DBMS_ASSERT** to validate SQL
- ✓ Use a **APEX-specific security tool** to help identify SQL Injection risks

A2:2017
Broken Authentication

Authentication

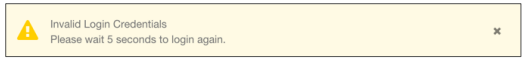
- In APEX, **Authentication** is the event when the **user provides a set of credentials** - typically a username & password - and they are verified or rejected by the corresponding **Authentication Scheme**
  - Result is a boolean
- From a technical point of view, it is **irrelevant as to how APEX arrives at the result**
  - Typically will be based on a valid username & password combination
  - But could be something as simple as “guess my number”

Authentication Schemes

- Out of the box, APEX can use the following **Authentication Schemes**:
  - Application Express Accounts
  - Database Users
  - HTTP Header Variable
  - LDAP
  - Open Door
  - Oracle Application Server Single Sign On
  - None
- Additionally, a **Custom** scheme - which can interface with almost anything - can be developed

Invalid Credentials

- All invalid authentication attempts are **logged**
- APEX can be configured to wait X seconds before allowing the next login attempt
  - Instance-level setting, applied to all applications



- Some authentication schemes only allow X number of invalid attempts before locking the account
  - Workspace-level setting

DEMO: Show wait time; enter invalid password and lock out

Session Duration

- A user's session will be valid until **any one of the following** occurs:
  - An **explicit logout** event occurs
    - Clicking Logout or Quitting the Browser
  - The user **manipulates the URL** and alters the Session ID portion
  - Either a **Session Duration** or **Session Idle Timeout** is reached
  - The **ORACLE_APEX_PURGE_SESSIONS** job runs
  - The user **alters** or **deletes** the corresponding session cookie
  - An APEX or Workspace administrator **manually purges sessions**

DEMO: Logout & try to rejoin; change session ID; delete cookies

Two Factor Authentication

- While **Two Factor Authentication** is **not a native feature** of APEX, it is **easy to implement** this via APEX
  - Create an application process that redirects to a TFA page if the user has not provided the correct code
- Almost any TFA mechanism is possible to integrate with
  - SMS
    - Plivo, Twilio, etc.
  - E-Mail
  - Google Authenticator

Checklist

- ✓ Ensure that all APEX applications use the **same authentication scheme**
- ✓ **Do not mix** Public & Internal Users in the same application
- ✓ Ensure that your directory is set to **lock accounts and require password changes**
- ✓ Consider **Two Factor Authentication** for more sensitive applications

A3:2017 Sensitive Data Exposure

Securing Data

- Data should be properly secured at the **lowest level possible**
  - Transparent to technology that accesses the data, as that will change over time
    - Oracle Forms > Oracle APEX > RESTful Web Service Calls
- The Oracle Database offers a number of features to do just this
  - Secure Views
  - Redaction
  - Virtual Private Database
  - Oracle Label Security

Secure Views

Secure Views

- Economy-Class Solution
  - Works in **any edition** of the database
  - Just as secure as **VPD & Redaction**; harder to maintain
- Allows us to **expose only the rows & columns** of data from a table that we want to
  - Most **URL Tampering** attempts will be **fruitless**, as only authorized data is displayed in the view
  - Combined with a shadow schema, it's easy to only expose a subset of rows & columns
  - Build APEX forms & reports against the view

Application Context

- Serves as a **secure data cache** for attribute-value pairs needed for fine-grained access control (secure views or VPD)
  - Cache eliminated the need to query the database to obtain this data, thus improving performance dramatically
- APEX provides a hook to set the **Application Context** on each page view

Incorporating Contexts Into Views

- We can retro-fit any view to incorporate an **Application Context** as part of their **WHERE** clause to filter which rows they return
  - For example: limiting which rows are returned based on which department a user is in
- If instrumented properly, the view will work both with and without APEX
  - Use `NVL(v('APP_USER'),USERNAME)` when evaluating the logged in user
  - This will default to the connected schema if the query is not coming from APEX

Virtual Private Database

Virtual Private Database

- VPD dynamically manipulates the **WHERE** clause of all queries against a specific table or view and applies a pre-determined condition
  - Does so without any modification to application code
- Ideal way to protect data, as it works regardless of how the data is accessed
  - SQL*Net, APEX, RESTful web services
- **No-cost feature** of Oracle Enterprise Edition Database
  - Not supported in XE, SE One & SE

Virtual Private Database

- For example:
  - `SELECT * FROM EMPLOYEES`
- After the VPD function is applied, dynamically & automatically becomes:
  - `SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID = 10`

Automatically added by the VPD Function

Unaltered Data

ID	Name	Department	SSN
1	Scott	10	111-11-1111
2	Brian	10	222-22-2222
3	Jack	20	333-33-3333
4	Anita	30	444-44-4444

Virtual Private Database

ID	Name	Department	SSN
1	Scott	10	111-11-1111
2	Brian	10	222-22-2222
3	Jack	20	333-33-3333
4	Anita	30	444-44-4444

Benefits of VPD

- **Secures data at the database layer**
  - Works regardless of the application or technology being used to access the table
  - Can be used with APEX **APP_USER** value to secure data from both APEX & SQL*Plus
    - `nvl(v('APP_USER'),USER)`
- **Simplifies development**
  - No sophisticated **WHERE** clauses need to be applied throughout the application
- Makes things like **URL Tampering irrelevant**
  - Simple “No Data Found” messages will be returned

Other Features of VPD

- **Column Relevance**
  - Policy applied only when a specific column is part of the SELECT clause
- **Column Filtering**
  - Only data in a specific column that are allowed by the policy are displayed; all other columns appear as NULLs
- **Application Context**
  - Set and use an Application Context for a more efficient VPD solution

OLS

Oracle Label Security

- **For-cost option** for Oracle Database EE
- Allows **each row to be classified**
  - Only users with the corresponding clearance can see those rows
- Helps **enforce regulatory compliance**
  - Ability to implement “need to know” access
- Integration with **Oracle Database Vault** & **Oracle Identity Management**

Oracle Label Security

```
SELECT * FROM EMP
```

User Label:
SENSITIVE

Name	Salary	Data Label	
~~SMITH~~	~~1000~~	~~Highly Sensitive~~	✘
JONES	1500	Sensitive	✔
KING	1250	Confidential	✔

Redaction

Redaction

- Oracle Data Redaction is a feature introduced in **Oracle Database 12c**
  - Also back-ported to 11.2.0.4
- Included as part of **Advanced Security Option** or ASO
  - List price is $15,000 per processor + support*
- Hides or "redacts" data automatically from user queries without any application modifications
  - For example - 123-45-6789 becomes XX-XXX-6789
- Source data remains unchanged

** as of 29-DEC-2015*

Redaction Use Case

- Redaction fits best where users need to **see any record, but not all sensitive information**
  - Call centers, hotels, airlines, etc.
- Part of the sensitive data can be used to help authenticate the user
  - "Last 4 digits of your Credit Card/SSN"
- There should be no way for the user to see the entire value of sensitive data
  - Thus reducing the likelihood of internal data theft

Unaltered Data

ID	Name	Department	SSN
1	Scott	10	111-11-1111
2	Brian	10	222-22-2222
3	Jack	20	333-33-3333
4	Anita	30	444-44-4444

Redaction

ID	Name	Department	SSN
1	Scott	10	XXX-XX-1111
2	Brian	10	XXX-XX-2222
3	Jack	20	XXX-XX-3333
4	Anita	30	XXX-XX-4444

Redaction Types

- **Full**
  - Redacts entire value and replaces with a space for VARCHARs, “0” for NUMBERs or “1-JAN-2001” for DATEs
- **Partial**
  - Redacts part of a value with a placeholder and displays a portion of the actual data
- **Regular Expression**
  - Uses a Regular Expression to filter data
- **Random**
  - Replaces characters with random equivalents

Redaction vs. Data Masking

- **Oracle Data Masking & Subsetting** is a for-cost feature for Oracle Enterprise Edition
  - List price is $11,500 per processor + support*
- Designed to change actual values of data from a production data set when it’s moved downstream to Dev/QA
  - Maintains the “shape” of the data
  - Updates it with random values
  - For example
    - “Scott” becomes “Rfsgo”
    - 012-34-5678 becomes 361-72-8427

** as of 29-DEC-2015*

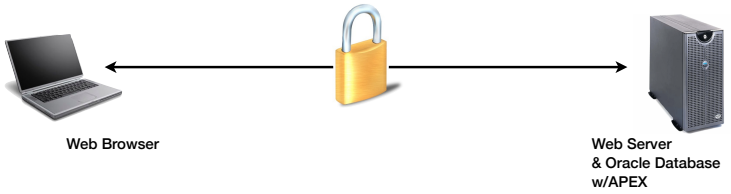
Redaction Warning

- Redaction is only applied to data as it is displayed
  - Not applied in the **WHERE** clause of a query
- Thus, use it only where there user will not have control over the **WHERE** clause
- Precautions to take in APEX
  - When using an IR & Redaction, disable options to filter the report for the redacted column(s)
  - Code change may be needed to exclude redacted column from **WHERE** clause of Classic Reports or other regions that the user can filter via input

HTTPS/TLS

HTTPS/TLS

- **TLS** (Transport Layer Security) is the replacement for **SSL** (Secure Sockets Layer)
- Encrypts all traffic between your web browser & the web listener



APEX HTTPS Options

- **Instance Level**
  - Secures your **APEX development environment**
  - Not always necessary for development
  - Necessary if you allow developers to log in to prod, as data queried in SQL Workshop needs to also be encrypted
- Three parameters to be concerned with:
  - **Require HTTPS**
  - **Require Outbound HTTPS**
  - **HTTP Response Headers**

APEX HTTPS Options

- **Application Level**
  - **Authentication Scheme > Cookie Attributes**
    - Should be called "**Require HTTPS for This Application**"
  - When set to **Yes**, APEX will not set session cookie if the application is run over HTTP
    - Thus no one will be able to login to your application

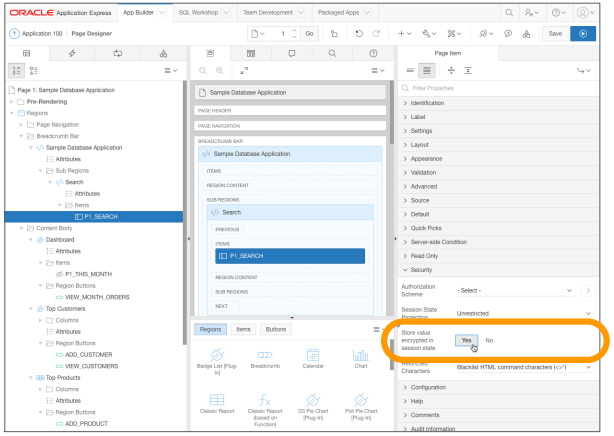
Item Encryption

Item Encryption

- APEX stores session state values in the database in clear text in the table **`WWV_FLOW_DATA`**
  - There is adequate security in place so that unauthorized users cannot see session state values from other sessions
- However, a **curious DBA or APEX administrator can view anyones session state**
  - Even if you do not want them to!

Item Encryption

- Can easily be configured on an **item-by-item basis**



No Encryption - Session State

Page Items

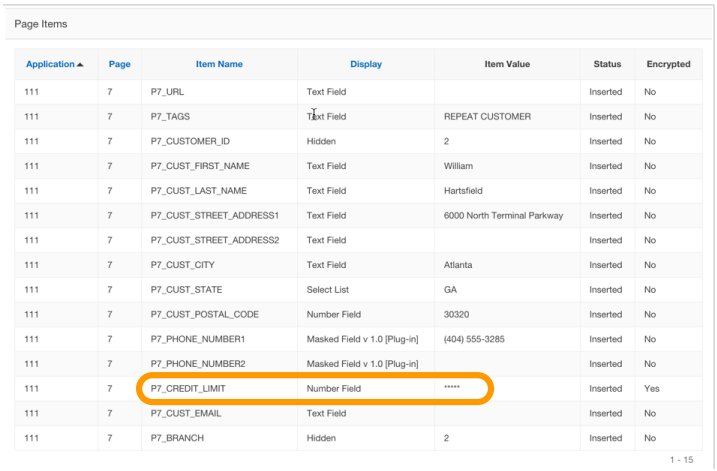
Application	Page	Item Name	Display	Item Value	Status	Encrypted
111	7	P7_URL	Text Field	http://www.johndulles.com	Inserted	No
111	7	P7_TAGS	Text Field		Inserted	No
111	7	P7_CUSTOMER_ID	Hidden	1	Inserted	No
111	7	P7_CUST_FIRST_NAME	Text Field	John	Inserted	No
111	7	P7_CUST_LAST_NAME	Text Field	Dulles	Inserted	No
111	7	P7_CUST_STREET_ADDRESS1	Text Field	45020 Aviation Drive	Inserted	No
111	7	P7_CUST_STREET_ADDRESS2	Text Field		Inserted	No
111	7	P7_CUST_CITY	Text Field	Sterling	Inserted	No
111	7	P7_CUST_STATE	Select List	VA	Inserted	No
111	7	P7_CUST_POSTAL_CODE	Number Field	20166	Inserted	No
111	7	P7_PHONE_NUMBER1	Masked Field v 1.0 [Plug-in]	(703) 555-2143	Inserted	No
111	7	P7_PHONE_NUMBER2	Masked Field v 1.0 [Plug-in]	(703) 555-8967	Inserted	No
111	7	P7_CREDIT_LIMIT	Number Field	1000	Inserted	No
111	7	P7_CUST_EMAIL	Text Field	john.dulles@email.com	Inserted	No
111	7	P7_BRANCH	Hidden	2	Inserted	No

No Encryption - SQL*Plus

```
  1  SELECT
  2    flow_id application_id,
  3    item_name,
  4    is_encrypted,
  5    item_value
  6  FROM
  7    wwv_flow_data
  8  WHERE
  9*   item_name = 'P3_SAL'
SQL> /

APPLICATION_ID ITEM_NAME       IS_ENCRYPTED  ITEM_VALUE
-------------- --------------- ------------- ----------------
           136 P3_SAL          N              5000
```

Encryption - Session State



Page Items

Application	Page	Item Name	Display	Item Value	Status	Encrypted
111	7	P7_URL	Text Field		Inserted	No
111	7	P7_TAGS	Text Field	REPEAT CUSTOMER	Inserted	No
111	7	P7_CUSTOMER_ID	Hidden	2	Inserted	No
111	7	P7_CUST_FIRST_NAME	Text Field	William	Inserted	No
111	7	P7_CUST_LAST_NAME	Text Field	Hartsfield	Inserted	No
111	7	P7_CUST_STREET_ADDRESS1	Text Field	6000 North Terminal Parkway	Inserted	No
111	7	P7_CUST_STREET_ADDRESS2	Text Field		Inserted	No
111	7	P7_CUST_CITY	Text Field	Atlanta	Inserted	No
111	7	P7_CUST_STATE	Select List	GA	Inserted	No
111	7	P7_CUST_POSTAL_CODE	Number Field	30320	Inserted	No
111	7	P7_PHONE_NUMBER1	Masked Field v 1.0 [Plug-in]	(404) 555-3285	Inserted	No
111	7	P7_PHONE_NUMBER2	Masked Field v 1.0 [Plug-in]		Inserted	No
111	7	P7_CREDIT_LIMIT	Number Field	*****	Inserted	Yes
111	7	P7_CUST_EMAIL	Text Field		Inserted	No
111	7	P7_BRANCH	Hidden	2	Inserted	No

1 - 15

Encryption - SQL*Plus

```
  1  SELECT
  2    flow_id application_id,
  3    item_name,
  4    is_encrypted,
  5    item_value
  6  FROM
  7    wwv_flow_data
  8  WHERE
  9*   item_name = 'P3_SAL'
SQL> /

APPLICATION_ID ITEM_NAME       IS_ENCRYPTED  ITEM_VALUE
-------------- --------------- ------------- ----------------
           136 P3_SAL          Y             9839BEFE425E74DX
                                             5C0318373DE67FCD
                                             C8B66BEF97B13AB3
```


Checklist

- ✓ Use **Secure Views, VPD or OLS** to limit what data a user can see
- ✓ Use **Redaction** where data needs to be seen but limited
- ✓ Always use **HTTPS**
- ✓ Be sure to **Encrypt** sensitive items
- ✓ Disable **Download** from sensitive reports

A5:2017
Broken Access Control

Authorization Schemes

- In APEX, **Authorization Schemes** determine what an Authenticated user can or can't see/access/execute based on some predefined condition
  - Result is boolean
- Source can be derived from:
  - SQL Query
  - PL/SQL Function
  - Item Value Comparison
  - Preference Value Comparison

Association

- Authorization Schemes **can be associated with every APEX component** - from the application itself to a page to a column in a report, and everything in between
- Best practice to create a "gatekeeper" scheme for each application
  - This scheme is associated with the application itself and only allows authorized users to use it

Integration

- There are APEX APIs available which allow easy integration of external user-to-role mappings to an APEX Authorization Scheme
- LDAP
  - **`APEX_LDAP.IS_MEMBER`**
- APEX Users
  - **`APEX_UTIL.GET_GROUPS_USER_BELONGS_TO`**
  - **`APEX_AUTHORIZATION.IS_AUTHORIZED`**
  - **`APEX_AUTHORIZATION.RESET_CACHE`**

Evaluation

- Authorization Schemes can be evaluated two different ways:
  - **Per Session**
    - Calculated once per session
  - **Per Page View**
    - Calculated once per page view and use for all components on the page
  - **Per Component**
    - Calculated for each individual component on each page
  - **Always**
    - Calculate for everything each page view
    - Least efficient, but ideal for testing

Reports

- **Batch Assign to Pages**
  - Allows quick & easy assignment of Authorization Schemes to all Pages
    - Application > Utilities > Cross Page Utilities > Grid Edit of All Pages
- **Authorization Scheme Utilization**
  - Displays which components are associated with which Authorization Schemes
    - Shared Components > Authorization Schemes > Utilization

Checklist

- ✓ Start adding Authorization Schemes at the **page level** and **work up from there**
  - Securing navigational controls - tabs, lists, buttons, etc. - is simply not enough, as users can easily manipulate the URL to access any page
- ✓ Use a **federated model** that manages access across all applications & all workspaces vs. stove-piping on a per-app basis
- ✓ Consider a **hybrid approach** (LDAP authentication, table-based authorization) when it is not possible to easily change user-to-role mappings in LDAP

A6:2017 Security Misconfiguration

Application Settings

- There are a number of **Application Settings** that can and should be changed to better secure your application from unauthorized access
  - These can all be found at either of the following:
    - **Shared Components > Edit Definition**
    - **Application Builder > Edit Application > Edit Application Properties**
- You will likely have **different options** set for **development vs. production**
  - Critical to ensure that the proper settings are set before deploying to production

Application Settings

- **Logging**
  - Useful for any instance, as this is what tells APEX to write to the APEX log tables
- **Debugging**
  - Should be disabled for Production
  - Can be programmatically enabled when necessary regardless of the value of this setting
  - Always enabled when running an application from the application development environment (4.2+)
- **Compatibility Mode**
  - Set to the most recent version of APEX

Application Settings

- **Availability**
  - Allows a developer to turn on or off a single application without having to turn off the web server
  - **Availability Status**
    - **Available** for production; any other for development
    - Some statuses can also have a **Message** or **Restricted User List**
  - **Build Status**
    - **Run Application Only** for production; **Run and Build Application** for development

Application Settings

- **Error Handling**
  - Specify an application-wide function to augment the reporting & presentation of errors generated in APEX
  - Functions must be in this format:

```
function my_function_name
  (
  p_error in apex_error.t_error
  )
  return apex_error.t_error_result
```

  - While not required, using a central error handling function is **strongly recommended**

Security Settings

- Each application also has a number of settings **specific to security attributes**
  - Unlike Application Settings, an application's Security Settings **rarely need to be changed** when moving an application into production
  - These can all be found at:
    Shared Components > Security Attributes

Security Settings

- **Authorization**
  - **Authorization Scheme**
    - Determines which Authorization Scheme a user must be a member of to access the application
    - If none required, should be set to **Must Not Be Public User**
  - **Run on Public Pages**
    - Determines whether or not the application-level Authorization Scheme is run on a **Public Page**

Security Settings

- **Session Timeout**
  - Determines the total duration of a session and the duration a session can be idle
  - Times are in seconds
- **Session State Protection**
  - Controls whether or not Session State Protection is enabled in your application
  - Should be set to **Enabled**
  - Enabling it is not enough: each page & item will also have to be properly configured for it to work

Security Settings

- **Browser Security**
  - **Cache**
    - Applications with sensitive data should set Cache to **Disabled**
      - This will change the page header to direct the browser to not cache pages from this specific application
    - HTTP server must support **cache-control** for this feature to work
  - **Embed in Frames**
    - Unless you have a specific need, should be set to Deny
    - HTTP server must support **X-Frame-Options** for this feature to work

Security Settings

- **Browser Security (cont)**
  - **HTML Escaping Mode**
    - Determines how APEX will escape characters when outputting data
    - **Basic**
      - &, ", < and >
    - **Extended**
      - &, ", <, >, ', / and non-ASCII characters if the character set of the database is not AL32UTF8

Security Settings

- **Rejoin Sessions**
  - As mentioned in Instance Settings, this option determines whether or not you can call an APEX URL without the Session ID
  - Should be disabled unless there is a specific need

Security Settings

- **Database Session**
  - **Initialization PL/SQL Code**
    - Formerly called **Virtual Private Database PL/SQL call to set security context**
    - Called at the earliest possible point when rendering/processing pages
    - Can be used for anything that needs to happen early
  - **Cleanup PL/SQL Code**
    - Called at the latest possible point when rendering/processing pages
    - Can be used for anything; close database links, unsetting contexts, etc.

Security Settings

- **Runtime API Usage**
  - Determine if and which APEX APIs this application can call that can:
    - Alter the application itself
    - Alter other applications in the workspace
    - Alter the workspace repository
  - All three should be disabled unless there is a specific need

Checklist

✓ Application Settings
- Ensure **Logging** is **enabled**
- Ensure **Debugging** is **disabled**
- Set **Compatibility Mode** to **most recent version**
- Set **Build Status** to **Run Only**
- Create an incorporate an **Error Handling** function

Checklist

✓ Security Settings

- Set an application-level **Authorization Scheme**
- Configure **Session Timeout and Idle Timeout**
- Enable and Configure **Session State Protection**
- Set **Allow Frames** to **Disabled** or **Same Site Only**
- Disable **Browser Cache**
- Set **Escaping Mode** to **Extended**
- Set **Rejoin Sessions** accordingly
- Disable **Runtime API** Usage

A7:2017 Cross-Site Scripting (XSS)

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

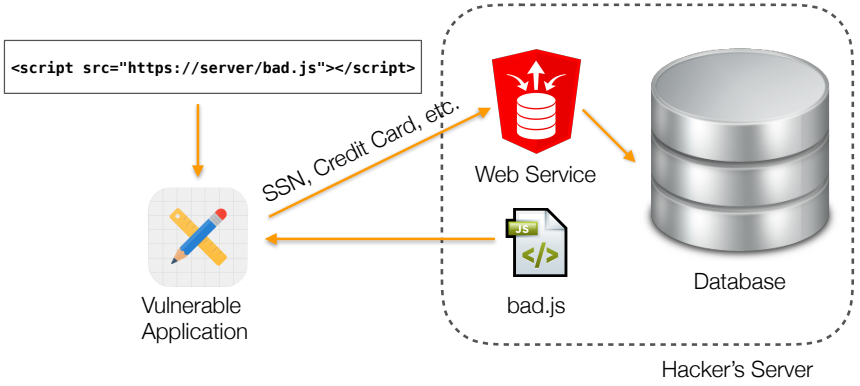
XSS in APEX

- **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



Restricting Input

- The range of valid characters **can be restricted** on an item-by-item basis
  - All Characters
  - Whitelist for a-Z, 0-9 and space
  - Blacklist HTML command characters (<>")
  - Blacklist &<>"/;,*|=% and —
  - Blacklist &<>"/;,*|=% or -- and new line
- Keep in mind that data in your application may originate where no such restrictions exist
  - Thus, always also escape when rendering

APEX_ESCAPE & Escaping Mode

- A new API, **`APEX_ESCAPE`** will return escaped versions of strings
  - More modern replacement for **`HTF.ESCAPE_SC`**
- What gets escaped when **`APEX_ESCAPE`** is called is controlled by an application's **HTML Escaping Mode**
  - **Standard**
    - **&, ", < and >**
  - **Extended**
    - **&, ", <, >, ', / and non-ASCII characters if the database character set is not AL32UTF8**

Checklist

- ✓ **Never disable escaping on columns**
  - When you do, be sure you know where the data is coming from or escape it with **APEX_ESCAPE**
- ✓ Be wary of **Application Items** that are rendered as HTML
  - Source is not escaped by default
- ✓ **Restrict characters** on input forms
  - Understand that not all input may come from your application
- ✓ Use an **APEX-specific security tool**
  - APEX-SERT or ApexSec

A8:2017 Insecure Deserialization

Insecure Deserialization

- If two **unlike systems** need to **interchange data**, then the data may have to be serialized before it's sent from one system to another

Insecure Deserialization

- **Serialization**
  - Converting an object into a format that can be transmitted from one system to another
  - Examples: A row of data to JSON or XML
- **Deserialization**
  - Converting that stream back into an object
  - Examples: Parse JSON and insert it into a table

Insecure Deserialization

- Both JavaScript & PL/SQL **provide native commands** to serialize and deserialize strings
  - **APEX_JSON** is also one of these
- Best practice is to use **vendor-provided or built-in parsers**
  - You can try to write your own
  - But why?

Example

- Web service allows the submission of data via JSON
  - Receives a JSON document with the user & role

```
{
"user" : "scott",
"role" : "user"
}
```

  - If a malicious user was able to modify the payload and submit a new JSON document, then they could potentially escalate privileges

```
{
"user" : "scott",
"role" : "admin"
}
```


Checklist

- ✓ Do not allow unauthorized sources to provide data
- ✓ Implement integrity checks
  - Such as a digital signature
- ✓ Build in validation logic at the API level
- ✓ Secure any web service and require authentication before allowing transactions to occur
- ✓ Use strict data typing
  - Reject & log data that does not conform
  - Not typically a problem in PL/SQL

A9:2017
Using Components with Known Vulnerabilities

Known Vulnerabilities

- When Oracle releases a patch, there are **two lists of things it addresses**
  - **Public Disclosures**
    - Can be seen in the release notes
  - **Private Disclosures**
    - Internal to Oracle
- In some cases, security vulnerabilities are on the Private list, and are not made public
  - Thus, it's best to keep as current as possible to ensure that as many security vulnerabilities are addressed

Known Vulnerabilities

- One of the best defenses is to **stay on the current release of APEX**
  - Or any software for that matter!
  - Oracle, ORDS, OS, Application Servers, etc.
- **Major upgrades can be painful** and require planning, testing & remediation
  - Example: 4.2 to 5.0
- **Minor upgrades** tend to be more **benign**
  - Example: 5.1.1 to 5.1.2
  - Still require planning & testing, but almost never remediation

Known Vulnerabilities

- Oracle's new versioning strategy is aimed to combat "**upgrade lag**"
- Smaller, more frequent releases should result in **easier, less lengthy upgrades**
  - APEX 18.1, 18.2 & 19.1 are all within a year
  - Previously, there was over a year between 5.0 and 5.1 and 5.1 and 18.1 each
- Stick to **built-in APEX components** as much as possible for easier upgrades
  - Built in components upgrade almost flawlessly
  - Custom code does not

Checklist

- ✓ Ensure that you're running the **latest release of Oracle APEX**
  - As well as any associated components
- ✓ Subscribe to and Apply Patches referenced in **Oracle Security Alerts**
- ✓ Keep an eye on industry publications for new exploits/ vulnerabilities

A10:2017 Insufficient Logging & Monitoring

Auditing

- APEX **does not** have any native Auditing capabilities built in
  - But that's OK, because the **Oracle Database does** and APEX can easily take advantage of them
- Some tools/features that can be used to audit include:
  - Database Triggers
  - Flashback Data Archive
  - Oracle Unified & Conditional Auditing
  - Oracle Audit Vault & Database Firewall

APEX Logs

APEX Logs

- APEX will automatically keep two logs:
  - **Page Views**
  - **Login Attempts**
- By default, APEX will only keep 2 weeks of data
  - Rotating between two log tables
- Interval can be modified as an APEX Instance Administrator
  - Not recommended to increase it much in high-volume systems, as there will be contention issues

Page Views

- All page views - full or partial - are logged via the view **APEX_WORKSPACE_ACTIVITY_LOG**
  - Can also be viewed via **Admin > Monitor Activity > Page Views**
  - Several “flavors” of the report; all of them based on the same data

Login Attempts

- All login attempts - successful or otherwise - are logged via the view **APEX_WORKSPACE_ACCESS_LOG**
  - Can also be viewed via **Admin > Monitor Activity > Login Attempts**

Monitoring the Logs

- It is **critical to monitor both of these logs** for anomalies
  - Excessive page views from a single user
  - Page views of invalid/non-existent pages
  - Odd user agents (such as sqlmap)
  - Excessive invalid login attempts
    - Same User
    - Wide Range of Users

Preserving the Logs

- Since APEX only keeps the logs for 2 weeks, it is **recommended that both logs** (page views & logins) are **copied to a more permanent place**
  - Can run a job nightly to copy yesterday’s data
  - You can’t get the data back one APEX purges it
- May be laws as to low long you can preserve the data and when you have to delete it

Flashback Data Archive

Flashback Data Archive

- Triggers **may introduce a high cost**
  - If they fire for each row in a large table, the data could be locked while the trigger performs the update
- **Memory consumption** may also be an issue with triggers
  - Especially those that that are fired for each update/insert

Flashback Data Archive

- **Flashback Data Archive** - aka **Oracle Total Recall** - provides the ability to **track & store all transactional changes** to a table over its lifetime
  - No longer need to use triggers or other constructs
  - More efficient and totally transparent
  - Compliant with record stage policies & audit reports
  - Requires Oracle 11.2.0.4+
  - No cost feature

FDA: Configuration

- Create a Flashback Archive
- Options include:
  - Tablespace
    - Where to store the data
  - Retention
    - How long to keep the data
  - Quota
    - How much space to use for the data

```
CREATE FLASHBACK ARCHIVE [archive_name]
  TABLESPACE [tablespace_name]
  RETENTION 1 YEAR
  QUOTA 100GB
```

FDA: Configuration

- Associate table with Flashback Archive
  - Must be **DBA** or have **FLASHBACK ARCHIVE ADMINISTRATOR** role to use

```
ALTER TABLE [table_name] FLASHBACK ARCHIVE [archive_name]
```

- Transactions on table will now be recorded
  - And retained as per the policy of the associated Flashback Archive

FDA: Usage

- Support for both **AS OF** and **VERSIONS BETWEEN** syntax

```
SELECT last_name, first_name, salary
  FROM EMPLOYEES
  AS OF TIMESTAMP TO_TIMESTAMP('2007-06-01 00:00:00',
    'YYYY-MM-DD HH24:MI:SS')
  WHERE employee_id=193;

SELECT last_name, first_name, salary
  FROM EMPLOYEES
  VERSIONS BETWEEN TIMESTAMP
    TO_TIMESTAMP('2007-06-01 00:00:00',
      'YYYY-MM-DD HH24:MI:SS')
    AND
    TO_TIMESTAMP('2009-06-01 00:00:00',
      'YYYY-MM-DD HH24:MI:SS')
  WHERE employee_id=193;
```


Unified & Conditional Auditing

Unified & Conditional Auditing

- Previous to Oracle Database12c, there were **several places** that the database stored **audit logs**:
  - **SYS.AUD$** - database audit trail
  - **SYS.FGA_LOG$** - fine-grained auditing
  - **DVSYS.AUDIT_TRAIL$** - Oracle Database Vault, Oracle Label Security
- A new feature in Oracle Database 12c - **Unified Auditing** - **consolidates all of these logs** into a single, unified log
  - **UNIFIED_AUDIT_TRAIL**

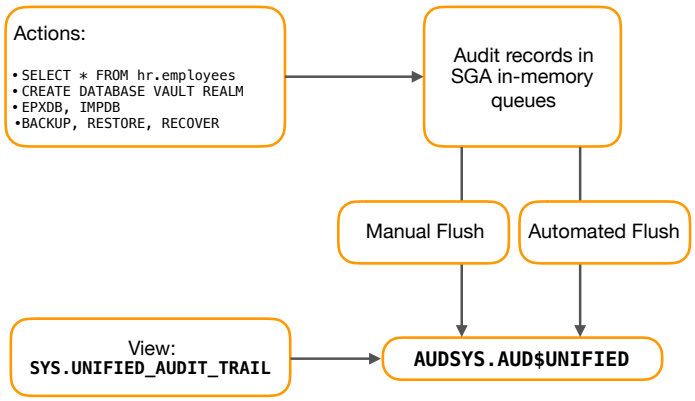
Unified & Conditional Auditing

- Unified & Conditional Auditing provides the ability to configure **precise, context-dependent logging**
  - Reduces the performance overhead associated with database auditing and enable more effective analysis of audit logs
- Can alter statements based on
  - **DDL or DML Type**
  - **Client IP/Location**
  - **Program**
  - **Time Period**

Unified & Conditional Auditing

- Unified Audit logs are stored in a new single-purpose schema - **AUDSYS**
- Existing audit data in the **AUD$** and **FGA_LOG$** - as well as all metadata and PL/SQL - will continue to reside in **SYS**

Unified & Conditional Auditing



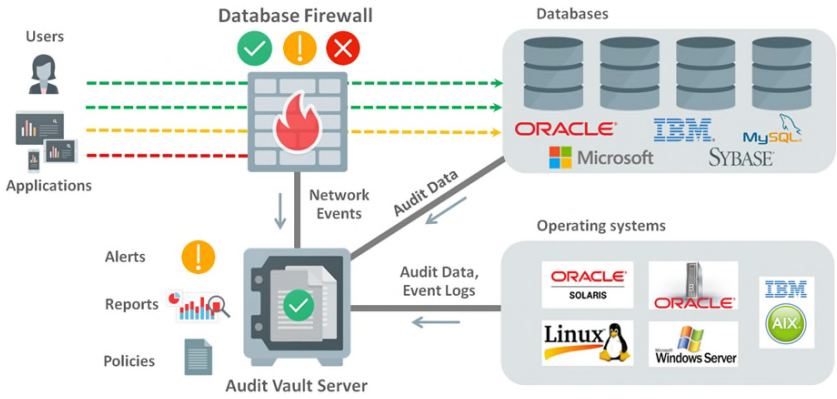
Oracle Audit Vault & Database Firewall

Oracle Audit Vault & Database Firewall

- Oracle Audit Vault and Database Firewall monitors Oracle and non-Oracle database traffic to detect and block threats, as well as improves compliance reporting by consolidating audit data from databases, operating systems, directories, and other sources.
  - Bundled together as a single product
    - Priced at $6,000 per processor*

** As of 18-JAN-2018*

Oracle Audit Vault & Database Firewall



Audit Vault

- Consolidates audit data into a single repository from multiple databases:
  - Oracle
  - MS SQL Server
  - IBM DB/2
  - Sybase
- Resulting repository can be reported on centrally and securely
  - Alerts can also be configured and sent based on user-defined events

Database Firewall

- Provides SQL grammar analysis engine that inspects SQL statements going to the database
  - Determines whether to allow, log, alert, substitute, or block the SQL entirely
- Support for multiply policy types
  - White list
  - Black list
  - Exception list
- Installed on the network on a bridge where it scans SQL traffic for suspicious payloads

Checklist

- ✓ At a minimum, **inspect the APEX logs regularly**
  - **APEX_WORKSPACE_ACTIVITY_LOG**
  - **APEX_WORKSPACE_ACCESS_LOG**
- ✓ Consider **backing up both tables** regularly
- ✓ **Flashback Data Archive** or **Unified Auditing** are good solutions for auditing needs
- ✓ **Audit Vault** & **Database Firewall** can provide additional protection

Summary

Summary

- The **OWASP Top 10 Threats are Real**
  - Based on tons of industry knowledge & expertise
- APEX applications are **largely secure**, but **can be susceptible** to any of these threats
  - Typically when a developer does something stupid
- **Security starts on day 1 and never ends**