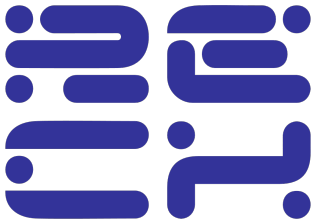
APEX Security - Preventing unauthorized access to your data



Recx

APEX Connect 2015

Recx and APEX Security

- APEX Security Consultancy for UK Gov. since 2009.
- Developing ApexSec product since 2010.
- Continual APEX Security Research (platform, samples, public applications).
- Presenting at various OUG conferences in UK since 2012.
- Published "Hands-On Oracle Application Express Security: Building Secure APEX Applications" in 2013.
- Presented at ODTUG KScope13 in New Orleans.
- Presented at APEX Forum in Vienna, 2014.
- Sponsored APEX World in Netherlands, 2014.
- Presented at ODTUG KScope14 in Seattle.

Recx and APEX Vulnerabilities

- SQL Injection - `wwv_flow_utilities.gen_popup_list` (APEX <3.2.1, CVE-2010-0892).
- SQL Injection - Application Builder 4000, (APEX <4.1, CVE-2011-3525).
- Cross-Site Scripting - Double tagging (APEX <4.1.1, CVE-2012-1708).
- Item Protection - Bypass vulnerability (APEX <4.2.1, CVE-2013-1519).
- ORDS URL Restriction Bypass (ORDS <2.0.8).
- SQL Workshop Privilege Escalation (APEX <4.2.6, CVE-2014-6483).

"We would like to recognise and thank Recx Ltd. for the use of their ApexSec analysis engine, which has been used to improve the security of Oracle Application Express..." - Oracle

Access-Control

In this session we are going to look at access-control - this term covers the following areas within APEX applications:

- Apache/ORDS
- Authentication
- Authorisation
- Item Protection

Web Server & ORDS

- Enable SSL and only allow HTTPS connections.
- Strip down the web/application servers:
  - Remove features, legacy content, default files, admin interfaces...
- Web server URL restrictions:
  - Allow only /i/ and /ords/, consider mod_rewrite
- ORDS configuration:
  - Define inclusionList: f,p,z,wwv_*,apex*
- Database schema permissions:
  - Procedures with EXECUTE granted to PUBLIC can be accessed via the browser.

Authentication

Define an Authentication Scheme:

- Application Express Accounts - Used for smaller applications and during development.
- Custom - Most common; ensure code is reviewed, operates as expected, and items are protected.
- LDAP Directory - Widely used, set Username Escaping to Standard.
- HTTP Header Variable - Used in some SSO solutions; prevent direct HTTP access (spoof header).
- Oracle Application Server Single Sign-On - Never observed.

Not recommended:

- Database Accounts
- No Authentication
- Open Door Credentials

Authentication

Any custom authentication scheme should consider:

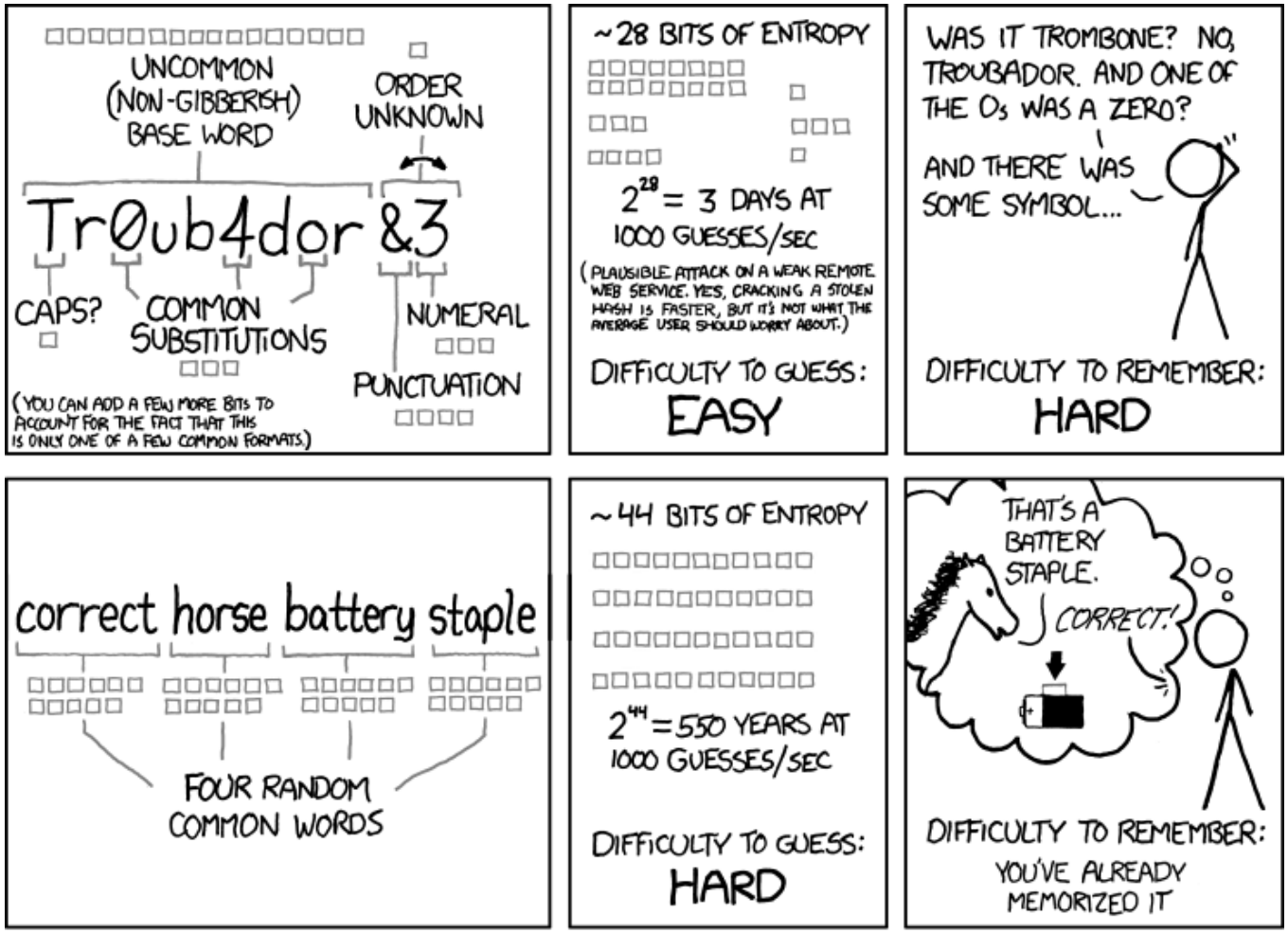
- Account signup and activation - where an application allows users to signup their identify should be verified, for example by sending an activation link to their email address.
- Account lockout - an account should be temporarily disabled after successive attempts to authentication with an incorrect password.

Authentication

Any custom authentication scheme should consider:

- Password storage - ensure user passwords are stored using a strong cryptographic hashing algorithm, salted with a user-specific value.
- Password complexity - depending on the sensitivity of data within the application and the privileges of the account, users must be forced to choose suitably complex passwords.
- Password age - users should change their passwords regularly.
- Password reset - when users forget their passwords, any reset mechanism must verify their identity (secret questions, email a reset link).
- Consider two-factor authentication for sensitive applications or transactions (token, SMS)

Passwords



Source: http://xkcd.com/936/

Authorisation

- Apply authorisation to APEX components to provide Role Based Access Control.
- APEX pages that are not linked to can still be accessed (sequential URL page numbering).
- Watch out for inconsistencies such as authorisation schemes applied to buttons and their associated processes.
  - Authorisation inconsistency demonstration...

Item Protection

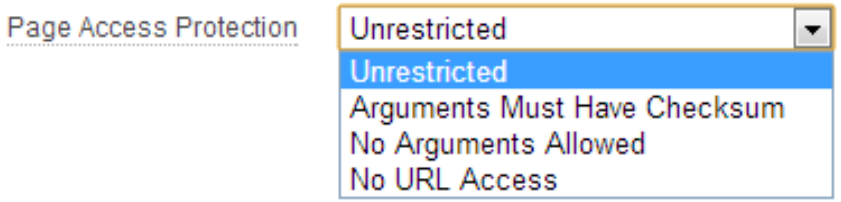
- What do we mean by “Item Protection"?
- Example of a common security risk.
- How Item Protection can defend against attacks.
- Correct use of Item Protection by classifying item types.

What is Item Protection?

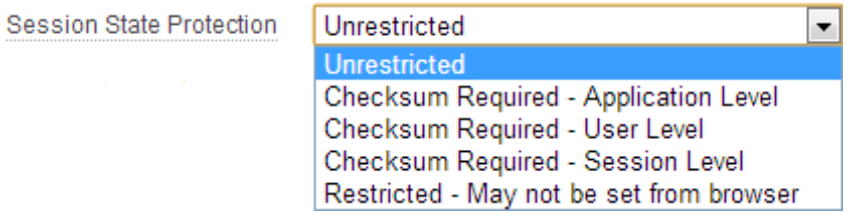
- APEX application items are defined at three levels: Global, Application or Page.
  - For example: F101_STATUS, P2_ID, ...
- Users of your APEX applications can change the values of items:
  - Form submission
  - On the URL
  - AJAX calls
- Protection exists to ensure values cannot be arbitrarily modified.

What is Item Protection?

- Page Access Protection (PAP)



- Session State Protection (SSP)



- Hidden Items (Value Protected)

Example Security Risk

Form on a Table with Report

Page: 51

Owner: APEXSEC

Table Name: EMP

* Select Column(s)

3.JOB (Varchar2)	1.EMPNO (Number)
4.MGR (Number)	2.ENAME (Varchar2)
5.HIREDATE (Date)	8.DEPTNO (Number)
6.SAL (Number)	
7.COMM (Number)	

Optional WHERE clause

```
deptno = 10
```

* Primary Key Type: Managed by Database (ROWID) / Select Primary Key Column(s)

* Primary Key Column 1: EMPNO

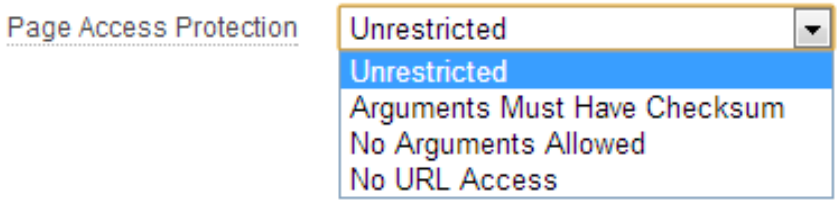
Primary Key Column 2: - Select Column -

	Empno	Ename	Deptno
	7839	KING	10
	7782	CLARK	10
	7934	MILLER	10

Example Security Risk

- When we clicked the edit link in the report we went to the Form page with a URL argument:
  - `f?p=123:5:0::::P5_EMPNO:7839`
- Changing the value of this item allowed us to access any record from the table, even when it was not presented in the original report.
  - `f?p=123:5:0::::P5_EMPNO:7566`
- Employee 7566 was in a different department.

Types of Item Protection

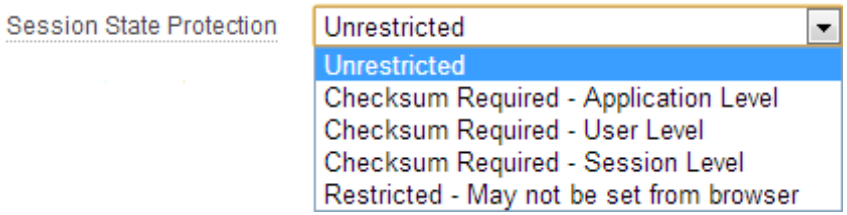


Page Access Protection:

- Protects the item from modification on the URL.
- Secured? Not at all. Item can be set:
  - URL (of another page)
  - Form submission
  - AJAX

Page Access Protection does not provide any security. It is required when Session State Protection is enabled, to ensure that links are generated with checksums.

Types of Item Protection



Session State Protection (Checksum):

- Protects the item from being set anywhere except the form containing the input widget.
- Secured? Almost!
  - Hidden items can be modified.

Types of Item Protection

Hidden Item “Value Protected":

- Protects the item from being modified in the HTML form at the point of form submission.
  - Set by default in APEX 4.0 and above.
  - Not set in older applications that are imported into new APEX instances.
- Secured? Only for Hidden items submitted in an HTML Form:
  - Hidden item can be modified via URL.
  - An AJAX POST can set ta Hidden item value.

Combining Item Protection Types

To secure our simple example we need to use all three protection mechanisms:

Page Access Protection:

- Page Access Protection is required to get a checksum parameter in the Edit link URL

Session State Protection:

- SSP protects the item from user modification everywhere (URL, AJAX) except the form

Hidden Item “Value Protected":

- Protect against modification of the HTML hidden input field in the Form page

Alternatives to Item Protection

In this specific example, we could secure the Report/Form combination without using Item Protection in several ways.

Match the WHERE clause in the Report and Form.

- But, linkage not apparent, need to keep in sync

Using the ROWID as the primary key

- But, is knowledge of valid ROWID secure enough?

Validations

- But, linkage not apparent...

Use them all! Good security is layered...

Classifying Items

Server-side logic items:

- Application - Set SSP to Restricted.

User-input items:

- Page - Set SSP to Checksum.

Display-only items:

- Page - Set SSP to Restricted.

Items to pass data between pages:

- Page (Hidden) - Set SSP to Checksum, set PAP of receiving page to Checksum, and set Value Protected to Yes.

Items modified by JavaScript:

- Page - No protection, take care when using item.
- Consider different name for unprotected items: JS123_USERNAME.

Problems with Item Protection

- APEX Temporary Storage Variables (g_x00..., g_f00...).
- Modifying items in JavaScript.
- Dynamic Actions that modify Display Only or Hidden fields.
- URLs constructed without PREPARE_URL.

Session State API Warning!

The SET_SESSION_STATE procedure should be used with caution. Passing an insecure item into p_name and p_value means your application has an Item Protection bypass vulnerability.

```
APEX_UTIL.SET_SESSION_STATE (
    p_name      IN    VARCHAR2 DEFAULT NULL,
    p_value     IN    VARCHAR2 DEFAULT NULL);
```

For example, we have seen this on demand process in a number of APEX applications:

Vulnerable

```
APEX_UTIL.SET_SESSION_STATE (
    wwv_flow.g_x02,
    wwv_flow.g_x03);
```

PREPARE_URL Warning!

The basis of item protection relies on the fact that a user cannot compute a valid checksum.
The PREPARE_URL function is used server-side to create links with correct checksums:

```
APEX_UTIL.PREPARE_URL (
    p_url           IN VARCHAR2,
    p_url_charset   IN VARCHAR2 default null,
    p_checksum_type IN VARCHAR2 default null)
RETURN VARCHAR2;
```

The ability to abuse PREPARE_URL requires control of an unprotected item in the URL *before* the name/value pairs:

Vulnerable

```
APEX_UTIL.PREPARE_URL (
    p_url => 'f?p='||:APP_ID ||':'||:GOTO_PAGE||':'||:APP_SESSION
        ||'::::P123_ID:666');
```

Summary

Web Server & ORDS:

- HTTPS, Remove features/content, URL filtering, inclusionList.

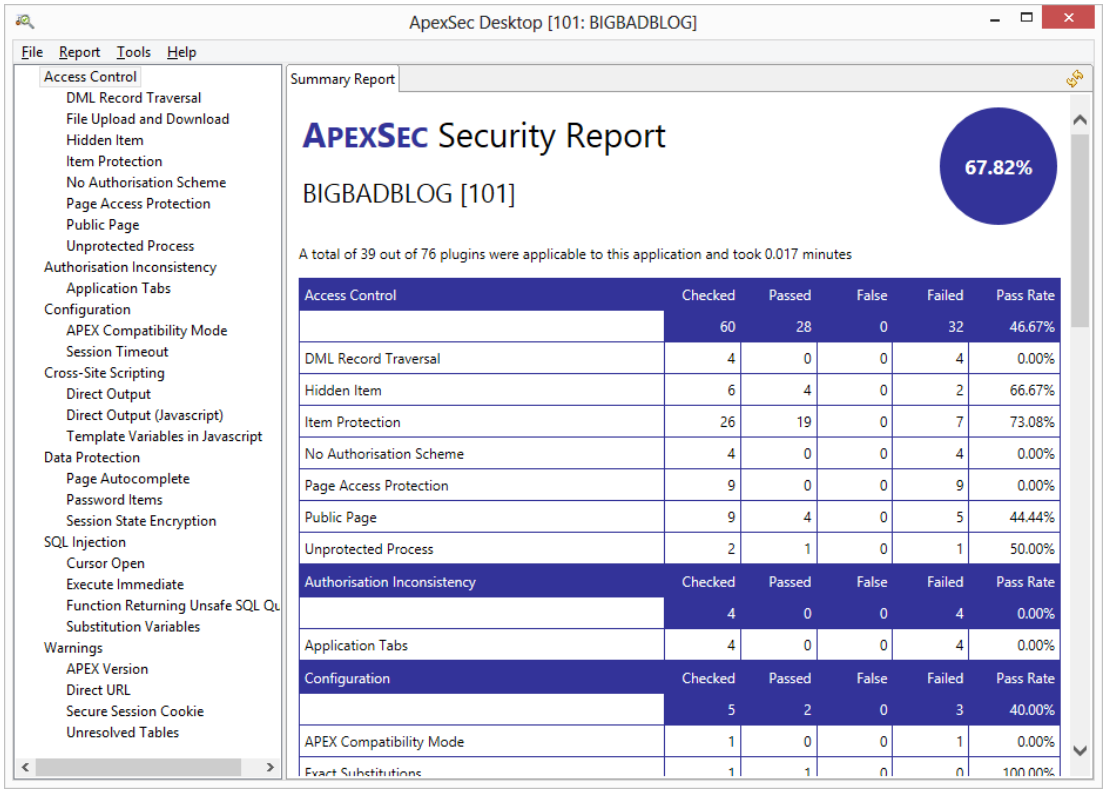
Authentication & Authorisation:

- Define an authentication scheme for the APEX application.
- When generating custom authentication schemes, follow best-practice for sign-up, activation, password policy. . .
- Use authorisation schemes to provide Role Base Access Control, and apply them consistently.

Item Protection:

- Item Protection is needed to protect items from modification.
- The different types of protection (PAP, SSP, Value Protected) must be used together to achieve real security.
- Most (if not all) items should be protected; an item with no protection should be the exception not the norm.

ApexSec performs an automated security analysis of APEX applications. Our unique knowledge and experience of securing real-world APEX applications is embedded into ApexSec, and the product is continually being revised to detect new threats.



Email us to arrange a free trial.