

DUGN
2023

P L A Y

A G A I N S T



VULNERABILITY

W I T H

APEX

Mirela Ardelean

Lino Schildenfeld



Mirela Ardelean

@Mirela_RoOUG



@LinoSchilde



- AUSOUG APEX committee chair
- NZ APEX meetup organizer
- Blogger
- Conference speaker
- APEX World Member of the Month
- Mentor and trainer



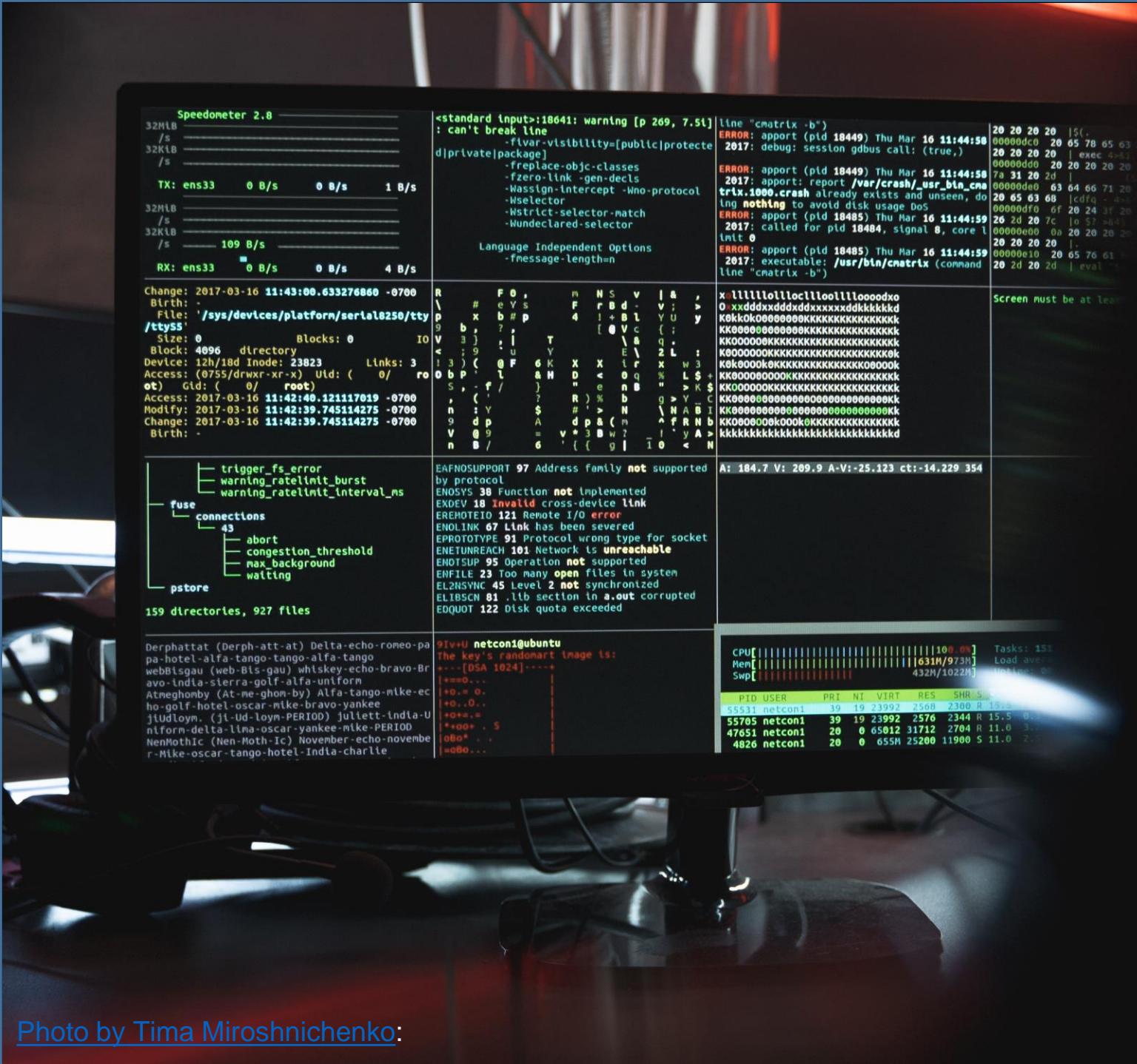
Mentor and Speaker Hub

Our goal is to *connect* speakers with mentors to assist in *preparing* technical sessions and *improving* presentation skills

Interested? Read more and get in touch

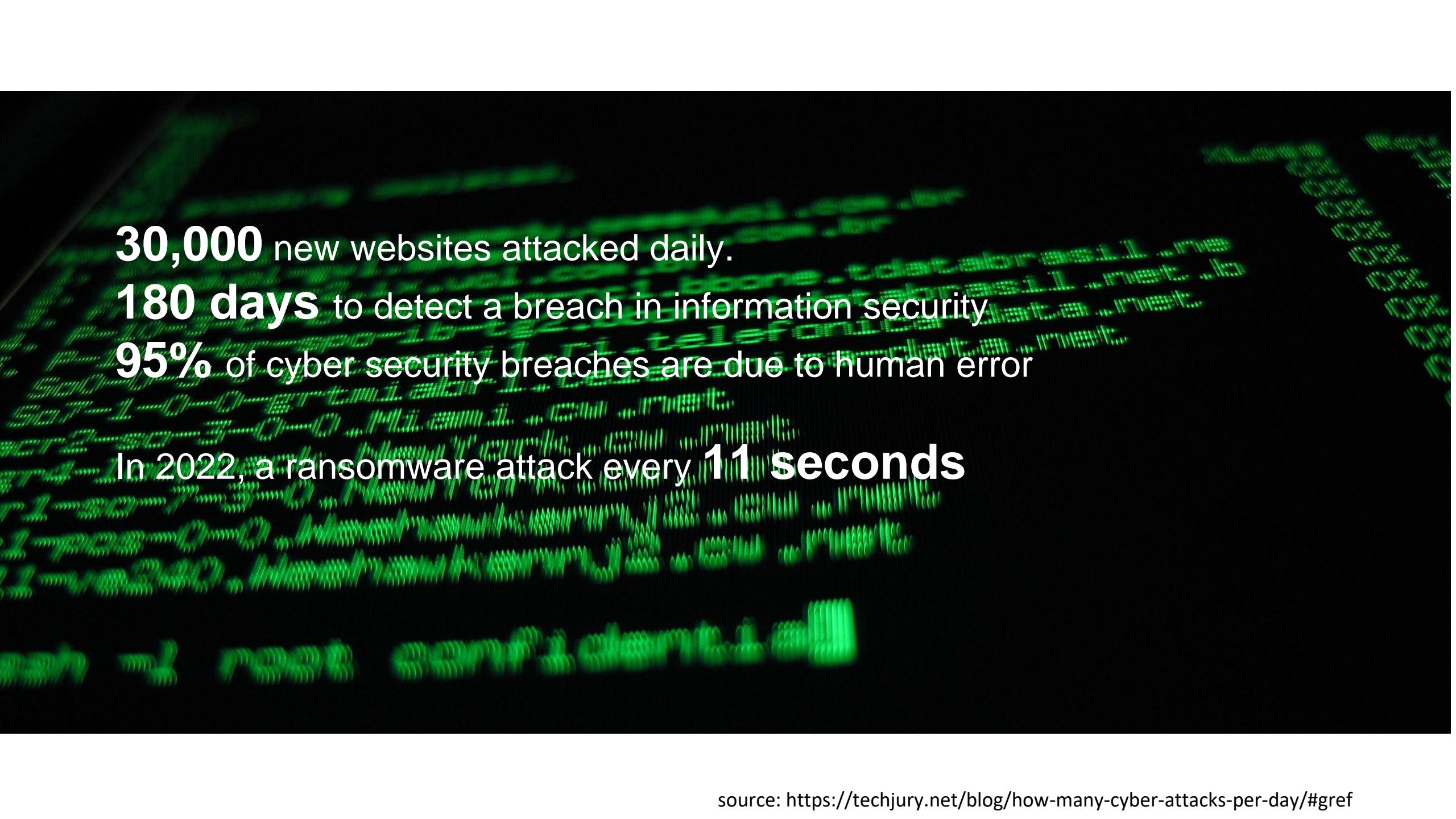
<https://mashprogram.wordpress.com>

Everything
connected to the
internet is a target
to hackers



39

seconds



30,000 new websites attacked daily.

180 days to detect a breach in information security

95% of cyber security breaches are due to human error

In 2022, a ransomware attack every **11 seconds**

Hacked Russian Taxi App Orders Dozens of Cabs to the Same Address, Chaos Ensues

The resulting frenzy reportedly caused hours of traffic delays.

BY PETER HOLDERITH SEP 2, 2022 3:23 PM

NEWS



CYBERCRIMINALS HACK COSTA RICA AND PERU, THREATEN LATIN AMERICAN GOVERNMENTS

17 May, 2022 Valentina Global



The international criminal organization Conti -based in Russia-, which specializes in extracting sensitive data from government agencies and attacking public platforms, is said to be behind the cyber-attacks suffered by the governments of Costa Rica and Peru, which have both countries on high alert.

data leaks - one of the primary
causes of data breaches

Data Leaks are usually caused by internal errors

A data breach is the intended outcome of a planned cyberattack

Social
Engineering

Misconfigured
Software
Settings

Recycled
Passwords

Causes of Data Leak

Physical Theft of
Sensitive Devices

Use of Default
Passwords

Software
Vulnerabilities



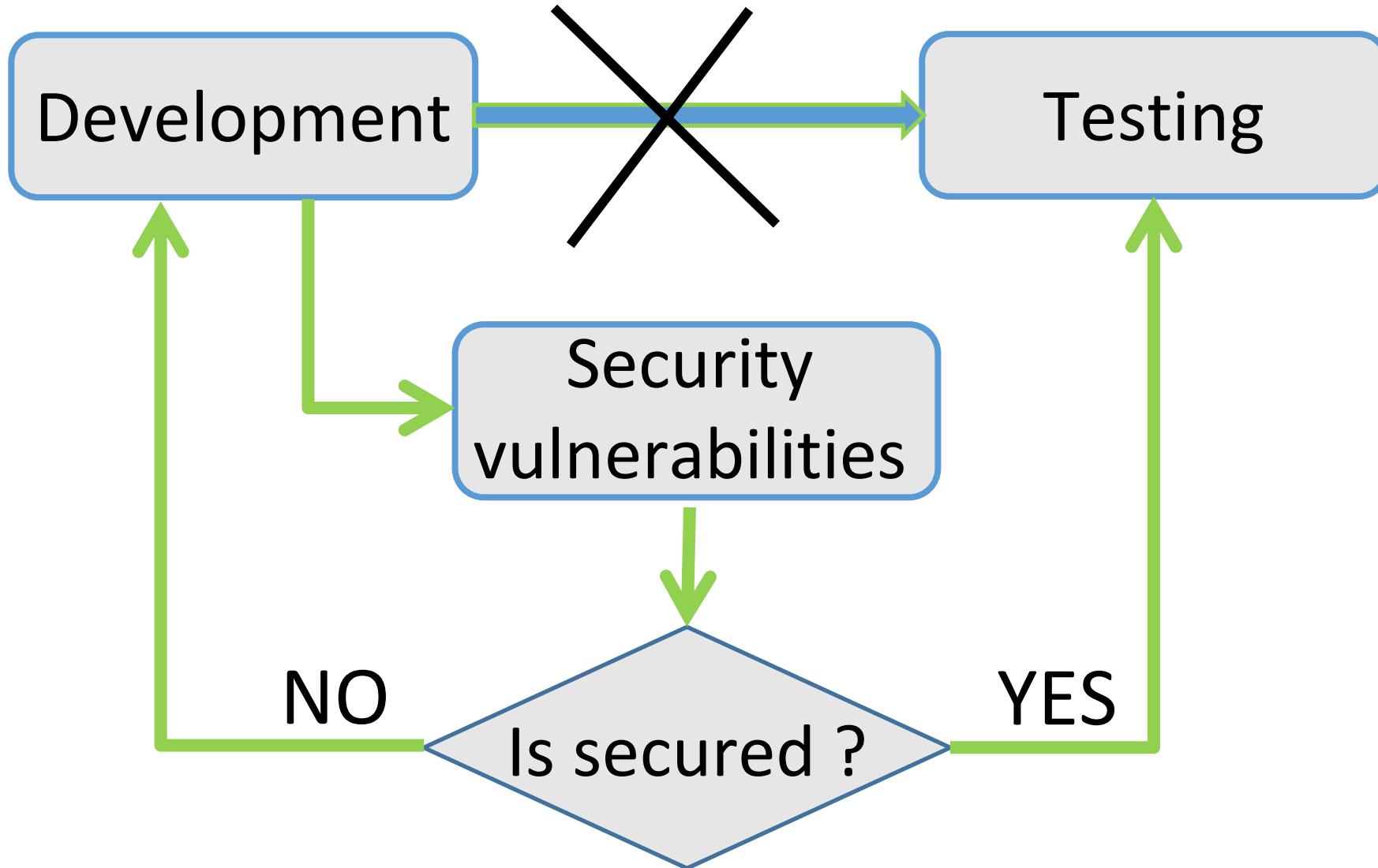
ā'pěks
(#orclapex)

**is
SAFE**

Are our APEX applications
secured ?

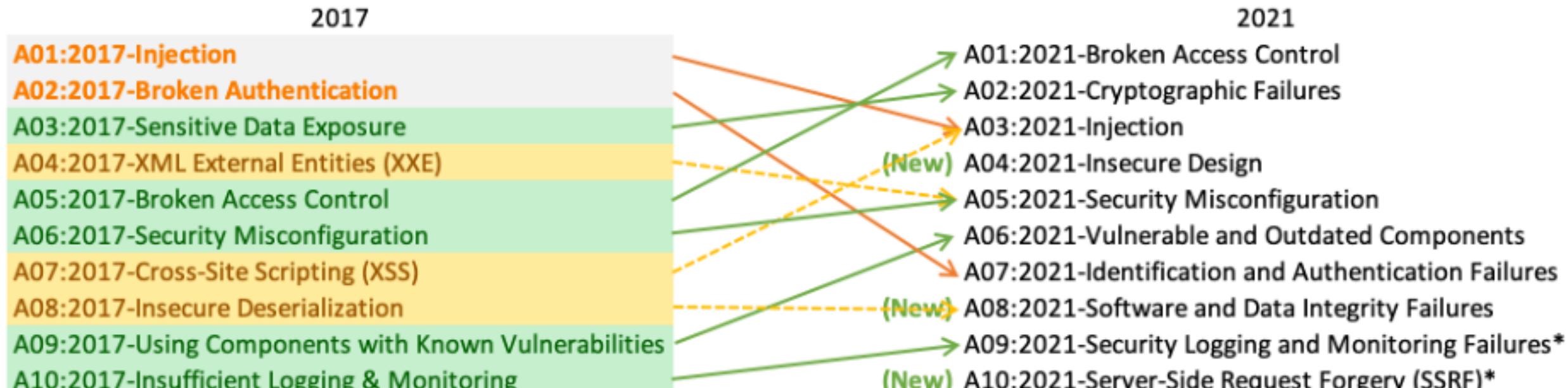
when to check
what to check
how to check

when to check



what to check

Top 10 Web Application Security Risks



* From the Survey

what to check

General

- Think of the security implications of your code
- **Architecture:**
 - Modular interface
 - Isolation of requests
 - Defense in depth protection against all kinds of attacks
 - Simplicity

Design

- Make **security-sensitive** parts of your code **simple** and **robust**.
- **Least privilege principle:** Don't require more privileges than necessary to run your code as the least privileged user necessary (use the `-S` flag). Make sure that the account on which you run your code has the minimum access and execute privileges that your code really needs; don't run it with admin privileges from your software;
- Choose **safe defaults**: For example, a random password rather than a standard default password;
- **Deny by default**: For example, when validating user input, expect the worst rather than trying to block known "bad" inputs;
- **Limit resource consumption**, to limit the likelihood of denial-of-service attack;
- **Fail securely**: For example, if there is a runtime error, assume she has none;
- In distributed or Web applications **don't trust** the user input, perform security checks or authentication on the server side; remember that HTTP response headers (like referer etc.) and HTTP query string values (from forged/manipulated) can be tampered with;
- **Cryptography:** **Use trusted, public algorithms** and don't invent your own cryptographic algorithms or protocols; reuse trusted code.

Coding

- **Don't trust input data:** Data coming from potentially malicious users is the single most common reason of security-related incidents (buffer overflow, SQL injection, Cross Site Scripting (XSS), code inside data etc.). Input data includes command-line arguments, configuration files (if accessible by not-trusted users), environment variables, cookies and POST/GET arguments etc;
- **Validate all input data:** Consider all input dangerous until proven valid, deny by default if not sure, validate at different levels, for example at input data entry point and before really using that data;
- **Don't make any assumptions about the environment:** make sure your code doesn't break with modified/malicious PATH, CLASSPATH and others environment variables, current directory, @INC Perl variable, umask, signals, open file descriptors etc;
- **Beware of race condition:** Can your code run parallel? what if someone executes two instances of your program at the same time, or changes environment in the middle of its execution?
- **Deal with errors and exceptions:** Don't assume that everything will work (especially file operations, system and network calls), catch exceptions, check result codes; don't display internal error messages, failing SQL query, stack trace etc;
- **Fail gracefully:** If there is an unexpected error that you can't recover from, then log details, alert the administrator, clean the system (delete temporary files, clear memory) and inform the user;
- **Protect passwords and secret information:** don't hard-code passwords (it's hard to change them and easy to disclose), use external files instead (possibly encrypted) or already existing credentials (like certificates), or simply ask the user for the password;
- **Be careful when handling files:** If you want to create it, report an error if it already exists; when you create it, set file permissions; if you open a file to read data, don't ask for write access; check if the file you open is not a link with the `lstat()` function (before and after opening the file); use absolute pathnames (for both commands and files); be extra careful when the filename (or part of it) comes from a user;
- **Temporary files:** Avoid them whenever possible. Pipes are a safer and more efficient way of communicating information between processes. If you really need temporary files, don't fall for the symbolic link attack (someone guesses the name of your temporary file, and creates a link from it to another file e.g. /bin/bash, that your program overwrites). Temporary files must have unique names that are hard to guess! (use `tmpfile()` for C/C++, `mktemp` shell command etc.);
- **Be careful with shell calls, eval functions etc.:** Such functions evaluate the string argument as code and interpret it, or run it on the shell. If an attacker managed to inject malicious input to that argument, you're executing his code.

Source:

https://security.web.cern.ch/recommendations/en/checklist_for_coders.shtml

how to check - Tools



APEX Sert



APEXSec



Adviser



APEX Project Eye



DIY Tool

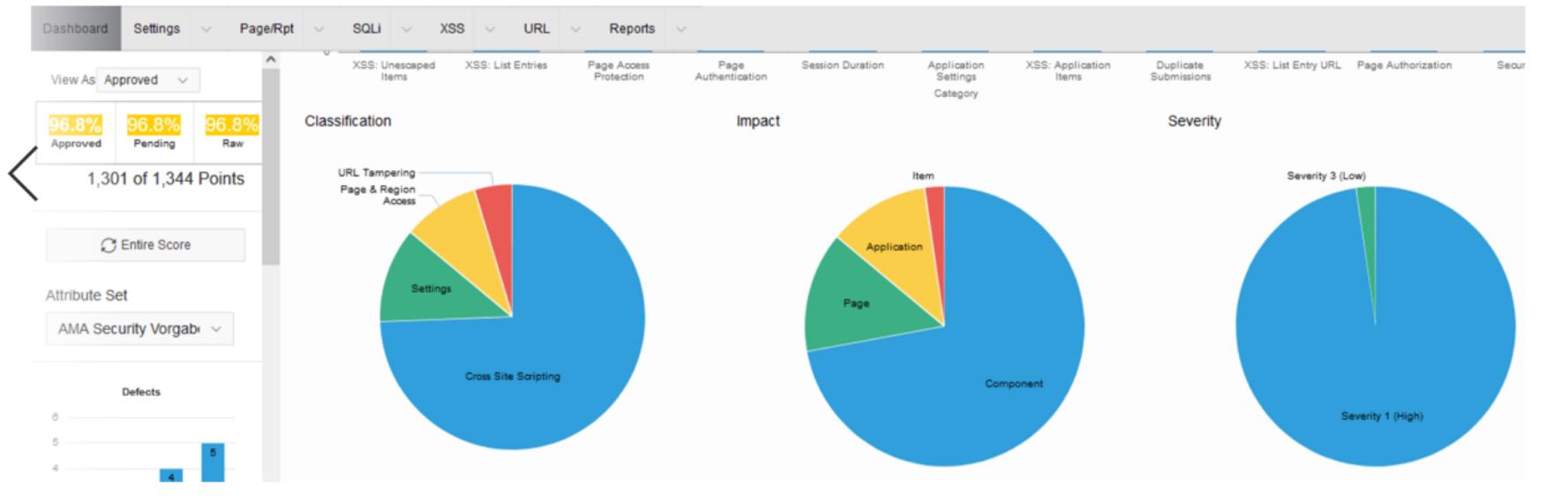


APEX Sert

APEX application that evaluates your APEX applications for security vulnerabilities

supports APEX 4.2, 5.0 & 5.1

<https://github.com/OraOpenSource/apex-sert>



XSS: Unescaped Output - Items

This table lists XSS: Unescaped Output items across three categories: Approved, Pending, and Raw. Each item includes its page name, item name, escape status, and result.

	Page Name	Page ↑	Item Name	Escape	Updated By	Updated On	Result	
1	Partitionen und Vorhaltezeiten	5	P5_MESSAGE	-	-	-	FAIL	+
2	Partitionen und Vorhaltezeiten	5	P5_MESSAGE_PROD_INSTANZ	-	-	-	FAIL	+
3	Session Control	14	P14_SID	-	-	-	FAIL	+
4	Session Control	14	P14_SERVER	-	-	-	FAIL	+



Categories

Category Name ↑	Classification
URL Tampering: Authorization Inconsistencies	URL Tampering
URL Tampering: Item Encryption	URL Tampering
URL Tampering: Item Protection	URL Tampering
URL Tampering: Page Access Protection	URL Tampering
SQL Injection: Reports	SQL Injection

Classification = 'URL Tampering'

Actions

Classification = 'SQL Injection'

Classification = 'Cross Site Scripting'

Classification = 'Page & Session Hijacking'

Classification = 'SQL Injection'

Classification = 'Cross Site Scripting'

Classification = 'Page & Session Hijacking'

Classification = 'Authorization Inconsistencies'

Classification = 'Item Encryption'

Classification = 'Item Protection'

Classification = 'Page Access Protection'

Classification = 'Reports'

All Categories

Home

Help

Fill

Cross Site Scripting

Page & Session Hijacking

SQL Injection

Event Listener

Settings

URL Tampering



APEX Sert

XSS: Unescaped Output - Items

SEV 1 ⚠ 🔧 ℹ Submit All ⚙️ 🖨️

Dashboard Settings Page/Rpt SQLi XSS URL Reports

View Approved As
Approved Pending Raw
96.8% 96.8% 96.8%

Approximate Time to Fix: .47 hours

Approved Pending Raw

3 out of 31 possible points 3 out of 31 possible points 3 out of 31 possible points

1,301 of 1,344 Points

Entire Score Page Score

Attribute Set AMA Security Vorgal

Defects 4 5

Page Name Page ↑ Item Name Escape Updated By Updated On Result

	Page Name	Page ↑	Item Name	Escape	Updated By	Updated On	Result		
🔗	Partitionen und Vorhaltezeiten	5	P5_MESSAGE	-	-	-	FAIL	+	💬
🔗	Partitionen und Vorhaltezeiten	5	P5_MESSAGE_PROD_INSTANZ	-	-	-	FAIL	+	💬
🔗	Session Control	14	P14_SID	-	-	-	FAIL	+	💬
🔗	Session Control	14	P14_SERVER	-	-	-	FAIL	+	💬

DEMO



APEX Sert



Lino Schilde
@LinoSchilde

Dear #orclapex friends, I stumbled across an application hidden under FR-2201 a few days ago :

Here is an upgraded version of APEX SERT [lschilde.blogspot.com/2023/03/apex-s...](http://lschilde.blogspot.com/2023/03/apex-sert.html)

Please post your feedback and any issue you might have.

12:07 AM · Mar 16, 2023 · 2,263 Views

<http://lschilde.blogspot.com/2023/03/apex-sert.html>



Lino Schilde @LinoSchilde · Apr 11

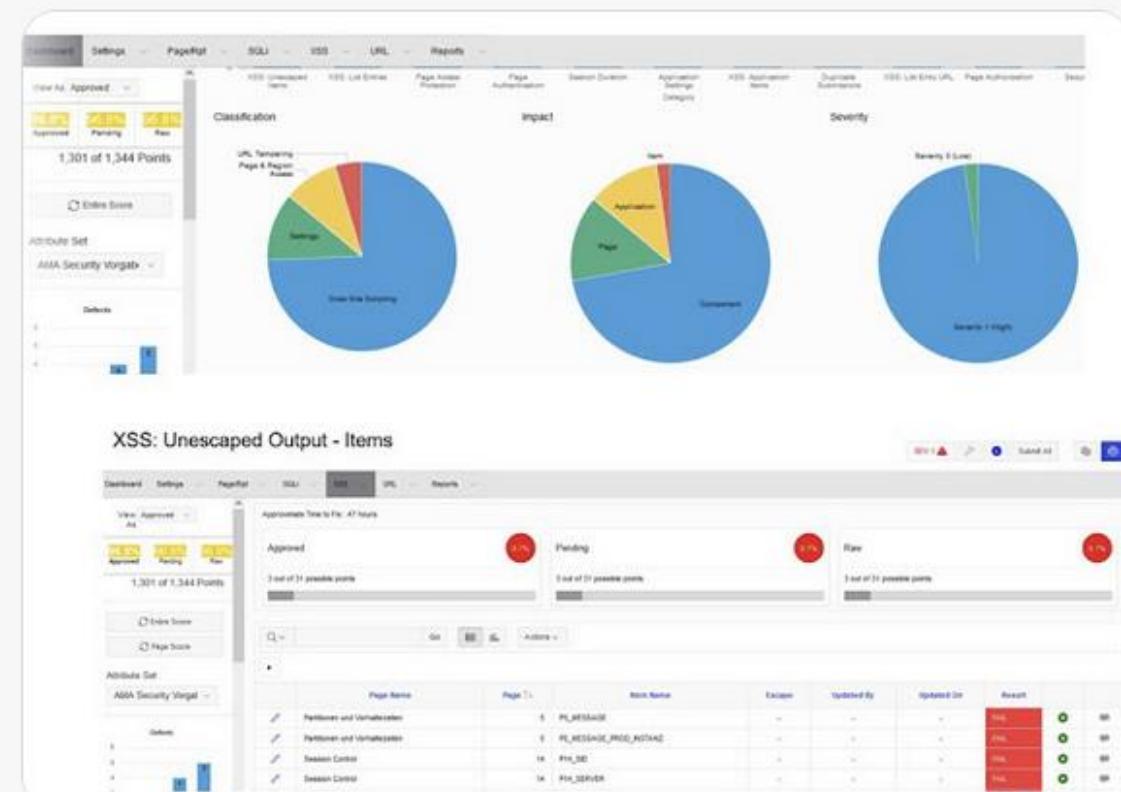
APEX SERT 22.2 is NOW available for download.

...

If you are running APEX 22.2 this is the version for you.

[lschilde.blogspot.com/2023/04/apex-s...](http://lschilde.blogspot.com/2023/04/apex-sert.html)

#orclapex #share #provideFeedback



6

28

58

4,980





APEXSec

PROS

- performs security scans of APEX applications and DB code
- Offline scanning
- It is a commercial product

Drag and Drop
Project File

ApexSec Desktop
File Report Tools Help

Welcome

Getting started with APEXSEC

To begin securing your Oracle APEX application, create a new project and specify how APEXSEC can access your application.

[Click here to create a New Project...](#)

Alternatively you can continue to work on an existing project file.



VEMACS Electronic Document Storage



APEX SECURITY221



APEX SECURITY



TT_BASELINE

<https://apexsec.recx.co.uk/>

Licensed to: Lino Schilde, Skillbuilders
Browser Version: Internet Explorer 11.0

APEXSEC desktop v3.1.24.142



ApexSec Desktop - APEX SECURITY [96792]

File Report Tools Help

Issues

Access Control

Hidden Item

Display and hidden Items [15]

Item Protection (Application)

Item Protection (Page)

No Authorisation Scheme

Page Access Protection

Public Page

Authorisation Inconsistency

Configuration

Cross-Site Scripting

Data Protection

SQL Injection

Dynamic SQL

SQL Injection [12]

Execute Immediate

Substitution Variables

Warnings

Issue Report

SQL Injection: Dynamic SQL

Page SQL Injection [12] (No Authorisation Required)

Item	Description	Detail
Page Process [Dynamic SQL - Demo 4]	Call to the function apex_collection.create_collection_from_query	p_query ()

Issue

A number of APEX procedures and functions require a parameter that represents a dynamic SQL Query in a string data type. SQL Injection conditions arise when user input is used in the construction of the dynamic query without appropriate validation or escaping.

Even when using bind variables or the V() function, SQL Injection can arise if the item is appended to the SQL query before being executed or returned. This is because the binding occurs at the time of concatenation, not at execution time.

For example, the following statements require dynamic SQL statements as parameters:

```
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY2
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY_B2
APEX_COLLECTION.MERGE_MEMBERS
APEX_ITEM.POPUP_FROM_QUERY
APEX_ITEM.POPUPKEY_FROM_QUERY
APEX_TTEM SELECT TST FROM QUERY
```

Page Process - Process Text

```
declare
l_query varchar2(2000);
begin
APEX_COLLECTION.DELETE_ALL_COLLECTIONS;
l_query := 'SELECT empno,ename,sal FROM emp WHERE job = '''|| :P12_INPUT4 || '''';
--FIX l_query := 'SELECT empno,ename,sal FROM emp WHERE job = v('''P12_INPUT4''')';
APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY(
  p_collection_name => 'EMP',
  p_query => l_query);
end ;
```



APEX Project Eye



APEX Project Eye (APE)
@APEXProjectEye

...

APEX-SERT quality assurance rules are now available to use with APEX Project Eye!

This rule set includes 100+ rules covering SQL injection, XSS, URL tampering, & much more. Helping you to create & maintain better quality, more secure [#orclapex](#) apps.

#lowcode #qualityassurance

The screenshot shows the APEX Project Eye interface with the following details:

- Quality Standards** section.
- APEX-SERT RULE SET** (Migrated Rules from APEX-SERT) listed.
- Updated: 8 seconds ago.
- Standard** view selected.
- Rules** table:
 - Filter: High Severity, Inactive rule, Medium Severity, Low Severity.
 - Column headers: Seq #, Rule, Severity, Active, Variables, Errors.
 - Rule Category: Security.
 - List of rules (Seq # 390 to 600):
 - SERT - Page Settings: Browser Cache
 - SERT - Page Settings: Deep Linking
 - SERT - Page Settings: Duplicate Submissions
 - SERT - Region Settings: Export Region Data
 - SERT - Page Settings: Form Autocomplete
 - SERT - Region Settings: Maximum Row Count
 - SERT - Page Settings: Page Requires Authentication
 - SERT - Page Settings: Authorization Scheme
 - SERT - Region Settings: REST Enabled
 - SERT - Page Settings: Region Sessions
 - SERT - SQL Injection: Authorization Scheme contains ITEM
 - SERT - SQL Injection: Authorization Scheme contains DBMS_SQL
 - SERT - SQL Injection: Authorization Scheme contains EXECUTE IMMEDIATE
 - SERT - SQL Injection: Branch contains ITEM, sysref
 - SERT - SQL Injection: Branch contains DBMS_SQL
 - SERT - SQL Injection: Branch contains EXECUTE IMMEDIATE
 - SERT - SQL Injection: Region source contains DBMS_SQL
 - SERT - SQL Injection: Region source contains EXECUTE IMMEDIATE

3:59 PM · Feb 28, 2023 · 5,108 Views



APEX Project Eye



Screenshot of the APEX Project Eye application interface:

The interface shows the following sections:

- Inspector**: Shows the URL <https://apex.apexmd.com/lords/tp=155.670.5686057036608>.
- Quality Assurance**: Active tab.
- Watchtower**: Tab.
- Quality Assurance | Assessments**: Sub-section.
- Rules repository**: Sub-section.
- Browse issues by**: Sub-section.
- By rule**, **By component**, **By severity**: Buttons.
- Rule view**: Sub-section.
- Electro Test Standard (16)**: Category list:
 - Best practice (15)
 - Consistency (0)
 - Error (0)
 - Security (1)
- Quality Standards**: Sub-section.
- Assessments**: Sub-section.
- Quality assessment**: Sub-section.
- Assessment details**: Sub-section.
- Issues (16)**: Sub-section.
- Search bar**: Includes a magnifying glass icon and a dropdown menu.
- Actions**: Sub-section.
- Severity**: High and Low buttons.
- Details**, **Component Type**, **Component Name**, **Rule**, **Additional Comment**, **Add Exception**, **Fix**: Headers for the Issues table.
- Issues Data**: Two rows of data:

Details	Component Type	Component Name	Rule	Additional Comment	Add Exception	Fix
Pages	Client Exception Report	Slow Page Performance				
Pages	Deal History	Page should have authorization scheme set-up				
- Component**: Sub-section.
- ET DEV SERVER**: Sub-section.
- Session**: Sub-section.
- Debug**: Sub-section.
- Quick Edit**: Sub-section.
- Customize**: Sub-section.
- Actions**: Sub-section.
- Manage exception list**: Button.
- Purge assessment from history**: Button.
- Configure your AOP Key**: Sub-section.



APEX Adviser

perform various sanity checks on your application: checks for errors, security issues, performance bottlenecks, usability and quality assurance.

**App Builder ->
Application XXXXX -
> Utilities**

Check Page(s) Perform Check

The Advisor performs various checks on your application, including programming errors, security issues, quality assurance, and other best practices. For the list of checks, see [Checks to Perform](#). Once executed, your previous settings will be recalled for the next use.

Specify a comma separated list of pages to check, or leave blank to check all pages.

Page(s) ?

Pages last edited within the last 7 days

Checks to Perform

Select All Deselect All

Errors: ?

- References with Substitution Syntax
- References with Column Syntax
- References with Bind Variable Syntax
- Declarative References of Application Items, Page Items, Columns or Interactive Report Filters
- Referenced Page Number Exists
- Is Valid SQL or PL/SQL Code
- Fetch, DML, MR* Processes are Valid
- Unconditional Branch before other Branches
- Referenced Button in When Button Pressed exists
- Button is not compatible with Dynamic Actions

Performance: ?

- V Function used in SQL Statements
- User Interface includes compatibility JavaScript

Usability: ?

- Target Page Authorization is also set for Current Component
- Associated Item or Column of Validations

Quality Assurance: ?

- Hardcoded Application ID
- Report has Default Order
- Page Item has Help Text
- Deprecated attribute values

Accessibility: ?

- Theme Style tested for accessibility
- Page has page title
- Region has Row Header
- Page item has label

Warnings: ?



Select * from < APEX Views>

- apex_applications**
- apex_application_static_files**
- apex_application_processes**
- apex_application_pages**
- apex_application_page_regions**
- apex_application_page_proc**
- apex_application_page_items**
- apex_application_page_buttons**
- apex_application_page_reg_cols**
- apex_application_page_rpt_cols**

- apex_appl_page_igs**
- apex_appl_page_ig_columns**
- apex_application_page_ir**
- apex_application_page_ir_col**
- apex_application_page_daActs**
- apex_application_list_entries**
- apex_application_lists**
- apex_application_lovs**
- apex_application_page_map_layers**



Code Editor - Code

↻ ⌂ | ⌂

```
1 alert("Entered value &P10_ITEM.");    CAUTION
2
3 alert("Entered value &P10_ITEM!JS.");
4
5 alert("Entered value " + $v('P10_ITEM'));
6
7 alert("Entered value " + apex.item("P10_ITEM").getValue());
8
9 alert("Entered value " + apex.items.P10_ITEM.value );
10
```

```
1  DECLARE
2    l_text CLOB;
3    l_email VARCHAR2(50);
4    l_name VARCHAR2(50);
5  BEGIN
6    SELECT 'dummy@demo.com' email, ' there!!!!!!' name into l_email, l_name
7    from dual;
8
9    --l_text := 'LAST MESSAGE: &P9_COMMENT..'|| utl_tcp.crlf;
10   l_text := 'LAST MESSAGE: '|| :P9_COMMENT || utl_tcp.crlf; CAUTION
11
12  l_text := l_text || ' Changed by &APP_USER.'|| utl_tcp.crlf;
13  l_text := l_text || ' Hello '|| l_name || utl_tcp.crlf;
14
15  htp.p(l_text);
16 END;
```

8 | l_text := 'LAST MESSAGE: '|| apex_escape.html(:P9_COMMENT) || utl_tcp.crlf;

Source

Form Region

- Select -



Type

Expression



Language

PL/SQL



PL/SQL Expression

```
apex_util.prepare_url('f?p=&APP_ID.:&APP_PAGE_ID.:&APP_SESSION.::::P9_MSG:&P9_MSG.')
```

CAUTION



```
1 declare
2 l_query varchar2(2000);
3 begin
4     APEX_COLLECTION.DELETE_ALL_COLLECTIONS;
5     l_query := 'SELECT empno,ename,sal FROM emp WHERE job = '''|| :P9_INPUT4 || '''';
6
7     APEX_COLLECTION.CREATE_COLLECTION_FROM_QUERY(
8         p_collection_name => 'EMP',
9         p_query => l_query);
10 end ;|
```

CAUTION

```
1 begin
2 if :P10_ITEM6 = :P10_ITEM6_1 then
3   http.p('<h1>They match!<h1>');
4 else
5   http.p('<h1>Ups sorry, ' || :P10_ITEM6 || ' does not match ' || :P10_ITEM6_1 || '.</h1>');
6
7 end if;
8 end;
```

CAUTION

secure programming

*a way of writing code in a software so
that it is protected from all kinds of
vulnerabilities, attacks or anything that
can cause harm to the software or the
system using it*

Coding

- **Don't trust input data:** Data coming from potentially malicious users is the single most common reason of security-related incidents (buffer overflow, SQL injection, Cross Site Scripting (XSS), code inside data etc.). Input data includes command-line arguments, configuration files (if accessible by not-trusted users), environment variables, cookies and POST/GET arguments etc;

Assume nothing

Believe no one

Confirm everything



Proper security configuration

Use Session State Protection

Hidden and Protected

Use Bind variable



ā'pěks
(#orclapex)

**is
SAFE**

What questions do you have?





mirela.ardelean@roouug.ro
Ischilde@gmail.com