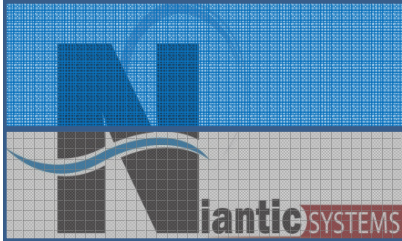


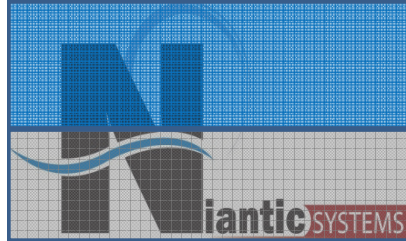
# Securing APEX Applications

Josh Millinger, President  
Niantic Systems, LLC



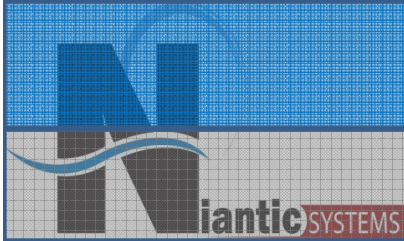
## Speaker Qualifications:

- Josh Millinger, Co-President, Niantic Systems, LLC
- CS degrees from UW-Madison, Johns Hopkins
- Former Oracle Sales Consultant and Founder of the Partner Technology Center
- 11+ Years of Oracle Web Development Experience
- Have Been Developing with and Teaching Apex Since Well Before It Was Even Released as a Product! Started with Excel Migration



# Niantic Systems

- Oracle Consulting with a Focus on Application Express
- Oracle Forms/Reports
- Discoverer
- Application Express Training
- Mentoring
- Customers in the Federal, Commercial, Healthcare, Higher Education, Construction verticals



# Securing Your Apex Applications

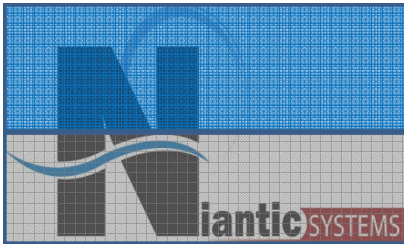
- Security Balance
- Create & Review Example Application
- Declaratively Secure Example Application
- Programmatic Measures
- Deployment Considerations
- Other Considerations

## How Secure is Secure Enough?

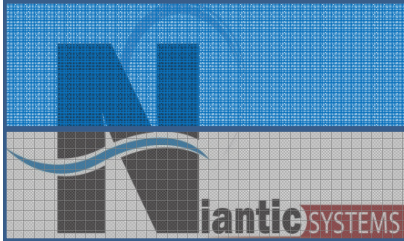
It depends on:

- What you're protecting
- Who you are protecting it from
- The likelihood of someone wanting to steal what you are protecting
- The repercussions you would face if someone were to successfully steal it

Unfortunately, adding Security is typically event-driven

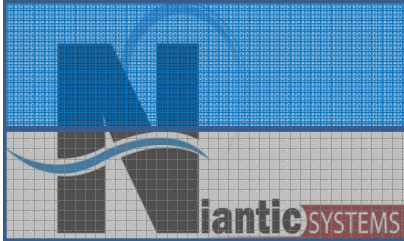


# Declarative & Programmatic Security Options



## Categories Defined

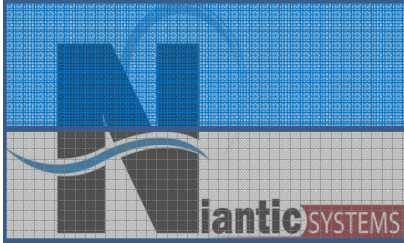
- \*Declarative – Little to no programming required
- Programmatic – Requires at least a small amount of coding by the developer



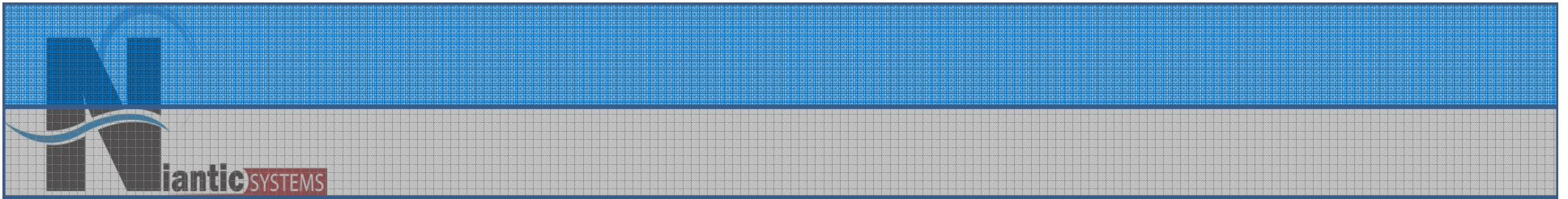
# Declarative Security Tools

- Authentication - Authentication Scheme Gallery
- Authorization – Access Control Page
- Session State Protection
- Deployment Build Status



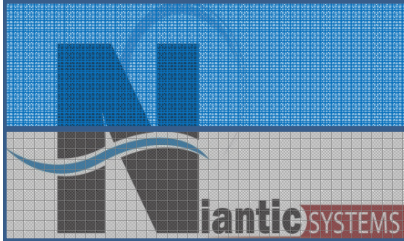


# Simple Application Example



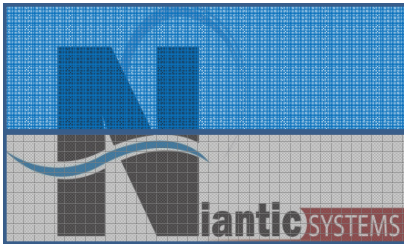
# User Authentication

Identifying the User

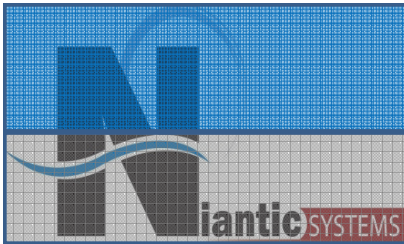


# Authentication Considerations

- Rarely Deploy with Apex Authentication
- Externally Managed Users e.g. Less Work(LDAP)
- Common Options in the Gallery
- Switching Schemes is Easy
  - Independent of Application
  - “Open Door” Can Be Handy
- Can Always Roll Your Own (Not Trivial)
  - LDAP vs Local Tables Example



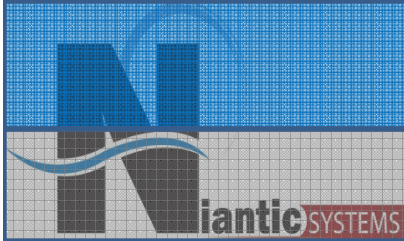
Q U I C K  
D E M O N S T R A T I O N  
Changing Authentication  
Schemes



# Access Control

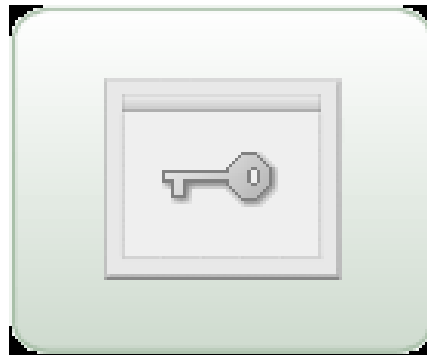
## Regulating Users Within the Application

- Managed through Authorization Schemes
- Schemes Applied Against Constructs
- Evaluation at Page or Session Level



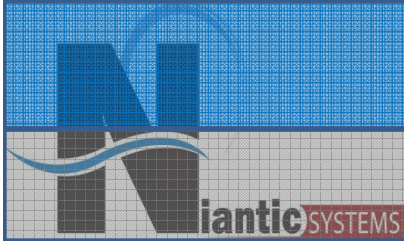
# Access Control Page Wizard

Regulating Users Within an Application the  
Easy Way



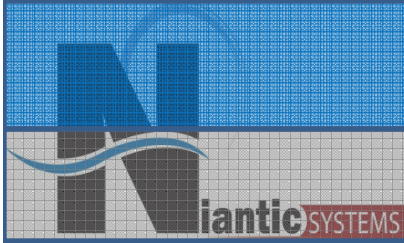
- Admin, Edit, & View Users
- Wizard Generated Components
  - Administration Page
  - Authorization Schemes
  - Two Tables in Application Schema
    - APEX\_ACCESS\_SETUP
    - APEX\_ACCESS\_CONTROL
  - Two Static LOVs





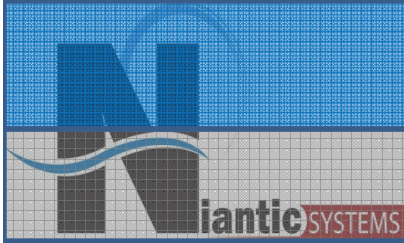
## Access Control Page, cont.

- Application Modes
- Independent of Authentication Schemes
- Applying the Authorization Schemes Still Requires Thought!



D E M O N S T R A T I O N

# Declarative Access Control



# Session State Protection

Consider this situation:

An unauthenticated malicious user visits your site. He notices that when he clicks on the View Details link for Content Items, he sees something like this:

**`http://hostname...:P2_CONTENT_ID:1`**

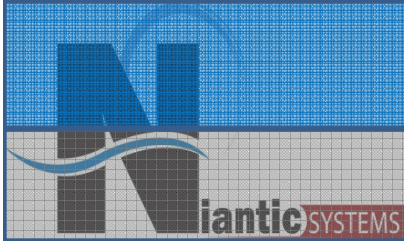
- So he clicks the link, and then changes the 1 to a 2, so the link now looks like this:

**`http://hostname...:P2_CONTENT_ID:2`**

- But that Content Item is restricted to Members Only, and the malicious user was able to view it!

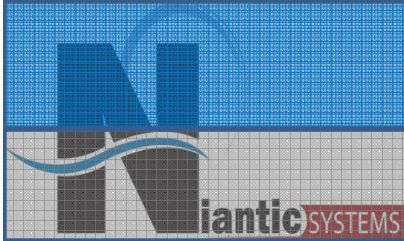
You have just fallen victim to one of the most popular form of hacking: URL Tampering

- Requires no programming skills
- ANYONE can learn how to do it
- And the results can be disastrous!



## Declarative Session State Protection

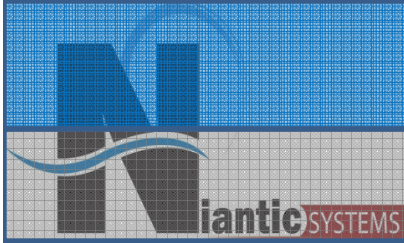
- Introduced in Apex 2.0
- How it works:
  - Rules applied to Items and Pages
  - Generates an additional Checksum and passes that as part of the URL
  - If the Checksum is absent or altered for protected item(s), the page will not render



## Declarative Session State Protection:

- Unrestricted
- Restricted
  - \*Application-Level Items
  - \*Display-Only Page-Level Items
- Checksum Required: Application Level
- Checksum Required: User Level
- Checksum Required: Session Level

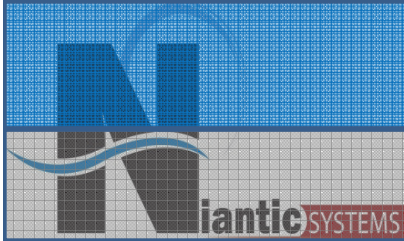




• D E M O N S T R A T I O N

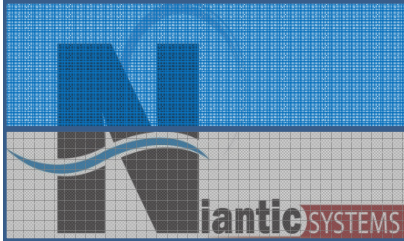
# Session State Protection

# Declarative Deployment Considerations



## Declarative Deployment Considerations

- Restrict Apex Access
- Build Status



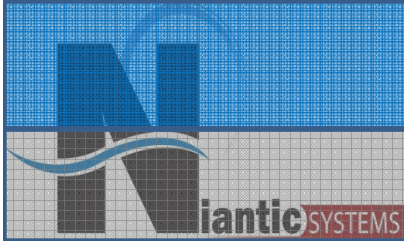
## Declarative Deployment Considerations

- Restrict Apex Access
  - Developer Access
  - Administrator Access

- A Q U I C K L O O K

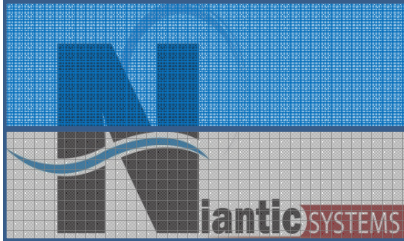
# Preventing Developer Access

```
BEGIN  
  WWV_FLOW_API.SET_SECURITY_GROUP_ID(  
    p_security_group_id=>10);  
  WWV_FLOW_PLATFORM.SET_PREFERENCE(  
    p_preference_name => 'DISABLE_ADMIN_LOGIN',  
    p_preference_value => 'N' );  
end;  
/
```



## Deployment Considerations, Cont.

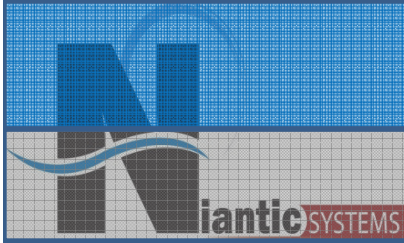
- Set Build Status to “Run Application Only” to disallow:
  - Debug
  - Trace
  - Developer Access



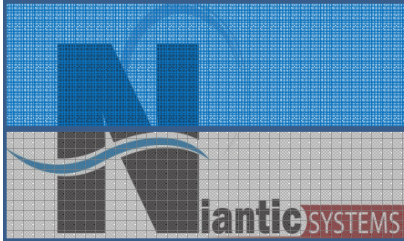
## 3.1 Runtime Engine

- 3.1 introduced new feature to “lock down” the Apex instance from development.
- By running scripts you can turn production instance into an entirely “Runtime” only instance that prevents all developer access





# Programmatic Security Considerations



## Programmatic Security Considerations

- SQL Injection Attacks
- Cross-Site Scripting Attacks
- VPD – Security starts at the database level

# SQL Injection Attacks

Consider a Simple PL/SQL Query Region...

```
DECLARE
    q varchar2(4000);
BEGIN
    q := 'SELECT *
        FROM tasks
        WHERE assigned = :APP_USER ';
    IF :P1_SEARCH is not NULL THEN
        q := q || ' AND category
            = ' || :P1_SEARCH ;
    END IF;
    return q;
END;
```

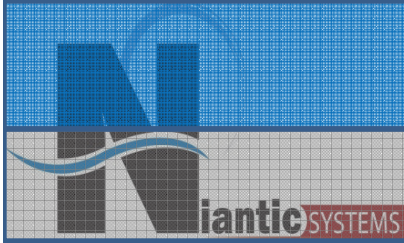
When a user provides “Email” for  
P1\_SEARCH  
our query will be...

```
SELECT *  
FROM tasks  
WHERE assigned=:APP_USER  
AND category = 'Email'
```

...but when a user provides “Email OR 1=1” for P1\_SEARCH our query becomes...

```
SELECT *  
FROM tasks  
WHERE ...  
AND category =  
'Email' or  
'a' = 'a'
```

...So ***never*** arbitrarily append user input into your application queries.

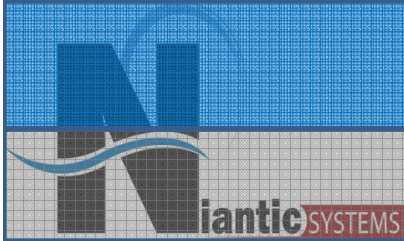


# Cross Site Scripting

Consider the following PL/SQL  
Region...

```
HTP.P ('The value of P1_ITEM is ' ||  
      &P1_ITEM.);
```





## Cross Site Scripting, cont.

A clever user could set  
P1\_ITEM with malicious  
javascript like so...

```
f?p=APP:PAGE:SESSION::::P1_ITEM:<some malicious  
javascript>
```

...which would cause our simple PL/SQL region to render the offending javascript directly on our page as...

**The value of P1\_ITEM is  
<some malicious jscript>**

- ***Never*** arbitrarily render user input to the browser!
- Use tools like **htf.escape\_sc** when possible to convert special characters...

```
HTP.P ( `The value of P1_ITEM is ` | |  
      htf.escape_sc (:P1_ITEM) );
```

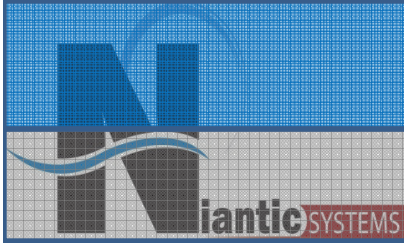
...so that our region would render  
as...

The value of P1\_ITEM is &lt;some malicious  
jscript&gt;

...which would successfully neutralize  
the attempted attack.

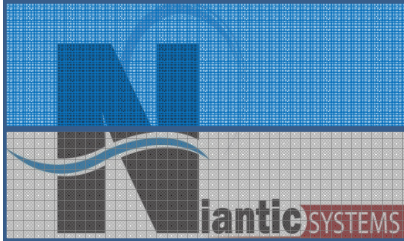
# Don't Forget the Database!

- Schema access may be possible from outside of your web application using ad-hoc query tools for example.
- VPD policies ensure access is properly controlled at the database level
- Leveraged from within Apex at application level



# DEMONSTRATION

# APEX & VPD



## More Extra-Application Measures

- Critical Patch Updates
- Operating System Security Patches/Updates
- Use Secure Sockets Layer (SSL)
- Password Policies (much like in Apex 3.0)

- Decide on a Security Balance
- Use the Right Tools for the Right Jobs
  - Declarative Options
  - Programmatic Techniques
  - Consider Other Means of Data Access



# Thank You!

- If you're so inclined, send me questions & comments directly:
  - Josh Millinger, Niantic Systems, LLC
  - Phone: 609.945.3151
  - Email: [jmillinger@nianticsystems.com](mailto:jmillinger@nianticsystems.com)

