Outline

- Preliminaries
- Motivation
- iLayer Framework
- Conclusion & Future Work
Content Management Systems (CMS)

- Online application that provides users the ability to easily create, design, publish and manage the content of a website
- Multiple users with varying roles
- Third party applications expand the capabilities and functionalities of content management systems
  - For example, a third party developed calendar application can provide schedule management

Image source: www.ubc.ca
Content Management Systems

- **Presentation Layer**
  - Displays to the visitor of the website the output (or content) of the CMS

- **Core Components**
  - Provides foundational CMS functionality

- **Database Layer**
  - Stores all content

- **Function Library / API**
  - Provides interface for integrating third party applications
Outline

- Preliminaries
- Motivation
- iLayer Framework
- Conclusion & Future Work
CMS Third Party Application Access Control

- Third party application (TPA) access control functionality is not well developed
- TPA’s typically have full administrator level access to the CMS and its content
- File permissions are the primary way to restrict access to TPA’s
  - Difficult to implement
    - CMS administrator must be able to translate access requirements of TPA’s into file permissions
    - Average CMS administrator may not have sufficient skill or experience
  - Implemented post installation
- More often than not, CMS administrators take minimal or no action to secure the CMS and its content from TPA’s
CMS Third Party Application Access Control

- Conducted study of popular CMS – Drupal
- Analyzed 412 third party applications’ database calls

<table>
<thead>
<tr>
<th>Table Name</th>
<th>Table Description</th>
<th>Potential Impact</th>
<th>% of 3rd Party Apps That Require Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>sessions</td>
<td>Contains user session information, e.g., userID, sessionID, user IP address, etc.</td>
<td>Session hijacking</td>
<td>2%</td>
</tr>
<tr>
<td>users_roles</td>
<td>Lists the assignments between users and roles</td>
<td>Privilege escalation</td>
<td>5%</td>
</tr>
<tr>
<td>node_revisions</td>
<td>Contains edits / revisions of node content</td>
<td>Content compromise</td>
<td>7%</td>
</tr>
<tr>
<td>permissions</td>
<td>Lists each user role’s permissions</td>
<td>Privilege escalation</td>
<td>7%</td>
</tr>
<tr>
<td>users</td>
<td>Contains usernames, passwords, profile information, etc.</td>
<td>Account compromise</td>
<td>23%</td>
</tr>
</tbody>
</table>

We believe that CMS users need additional tools and mechanisms to protect their online information from attacks via third party applications.
Outline

- Preliminaries
- Motivation
- iLayer Framework
- Conclusion & Future Work
iLayer – TPA Access Control Framework

- Manages user to third party application policies in Content Management Systems

Makes policy **recommendations** to CMS administrative user for third party applications.

Policies are **reviewed** and **set** by the CMS administrative user and **enforced** by the iLayer Framework.
iLayer Framework Overview

Core Components
- Content Management
- User Management
- Application Admin
- Session Management

Function Library / API
iLayer – Reference Monitor

Presentation Layer / Templates

Third Party Applications
Database Layer / Content
iLayer Policy
iLayer Framework Overview

Step A – iLayer Setup

CMS Library
core Function Refactoring

Determine Requested Access

Policy Rule Tuples with Recommendations

iLayer Policy

Approved All Selected Access

Step B – Third Party Application Installation

Manifest

Application Functions

Application core Functions

Application core Function Parsing and Replacement

Application i_core Functions

End

Deny All Access

CMS Admin

Step C – Runtime Enforcement (Reference Monitor enforces policy)
A. iLayer Setup – Policy Table Setup

- Create **iLayer Policy** table
- Policy is made up of three components:
  - **subject**: third party application that will be granted access
  - **object**: database table being given access to
  - **permission**: access privilege that is granted which could be either:
    - read (select)
    - write (delete, insert, update)
    - read & write
A. iLayer Setup – Refactoring

Library core Functions Refactoring

<table>
<thead>
<tr>
<th>Original core()</th>
<th>Refactored core() to i_core()</th>
</tr>
</thead>
</table>
| function core(arg) { … //extract table name and action from the arg … } | function i_core(3PA_Params, arg) { … //extract table name and action from the arg
| | //loop for all table names if(matchPolicy(3PA_Params, table, action) = null)
| | errorHandler();
| | else
| | core(arg);
| | }

The Laboratory of Information Integration, Security and Privacy – LIISP.uncc.edu
B. Third Party Application Installation

- **Determine Requested Access** by the third party application:
  1. **Manifest** provided by the third party developer
  2. **Application Access Analysis**

- **Policy Rule Tuples** presented to administrative user:
  - (subject, object, permissions)
    - (application_name, database_table_name, read)

```xml
<manifest>
  <policy_rule id="pr1">
    <subject>appName</subject>
    <object>birthday_table</object>
    <permission>select</permission>
    <required_flag>0</required_flag>
    <comments>Access is not required; but...</comments>
  </policy_rule>
  ...
  <policy_rule id="pr2">
    ...
  </policy_rule>
</manifest>
```
B. Third Party Application Installation

- Thumbs up or thumbs down **Policy Rule Recommendation** is presented
- Indicator of the community’s usage of the policy rule tuple

\[
x_i = \begin{cases} 
1 & \text{if } a_i \text{ is granted} \\
0 & \text{if otherwise}
\end{cases}
\]

<table>
<thead>
<tr>
<th>Application ID</th>
<th>Granted Accesses / a_i (object - permission)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sessions - read</td>
</tr>
<tr>
<td>001</td>
<td>0</td>
</tr>
<tr>
<td>002</td>
<td>0</td>
</tr>
<tr>
<td>003</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>412</td>
<td>0</td>
</tr>
<tr>
<td>413</td>
<td>0</td>
</tr>
</tbody>
</table>
B. Third Party Application Installation

- Thumbs up/down **Policy Rule Recommendation** is based on the maximum likelihood of the set of possible permission combinations for all requested objects based on historically granted accesses.

\[
X = \arg \max_{x_1, \ldots, x_{r-1}} P(x_r = 0, \ldots, x_n = 0 \mid x_1, \ldots, x_{r-1})
\]

| files - read | files - write | X       | P(R | X) | Recommendation |
|--------------|--------------|---------|-------|----------------|
| deny         | deny         | \{x_1 = 0, x_2 = 0\} | 0     | deny           |
| deny         | allow        | \{x_1 = 0, x_2 = 1\} | 0     | deny           |
| allow        | deny         | \{x_1 = 1, x_2 = 0\} | .2    | allow          |
| allow        | allow        | \{x_1 = 1, x_2 = 1\} | .5    | allow          |
B. Third Party Application Installation

- Number of conditional probability computations equal $2^n$, where $n$ equal the number of requested accesses

- Distribution of number of accesses for 412 Drupal TPA’s (modules) – average 2.45 and median 2.0
B. Third Party Application Installation

- All the policy rule tuples are presented to the CMS administrative user for a **Policy Decision:**
  - Approve all selected access
  - Deny all access

- **Function Parsing and Replacement**
  - Third party application code is parsed and all instances of `core` functions are replaced with their corresponding `i_core` functions
C. Runtime Enforcement

Step A – iLayer Setup

- CMS Library core Function Refactoring
- iLayer Policy

Step B – Third Party Application Installation

- Manifest
- Application core Functions
- Application Functions
  - Application core Function Parsing and Replacement
  - Application i_core Functions

- Determine Requested Access
- Application Access Analysis
- Policy Rule Tuples with Recommendations
- CMS Admin

Step C – Runtime Enforcement (Reference Monitor enforces policy)

- Approved All Selected Access
- Deny All Access
- End
Prototype – Drupal CMS Platform

- Limited proof of concept prototype
Outline

- Preliminaries
- Motivation
- iLayer Framework
- Conclusion & Future Work
Conclusion

- Presented a third party application access control framework for content management systems
- Provides policy setting functionality, including a policy setting recommendation capability
- Policies are reviewed and set by the CMS administrative user and enforced by the iLayer Framework
- Implemented a prototype of the iLayer Framework on the Drupal Content Management System
Future Work

- Expanding the current Drupal prototype from proof of concept to fully functional system
- Providing the ability to review and update third party application policies post installation
- Conducting a benchmark study measuring the performance impact of iLayer on a content management system
- Conducting a detailed survey and analysis of the threat vectors
- Conducting an extensive user study
Questions

Gorrell Cheek
University of North Carolina at Charlotte
gcheek@uncc.edu