Magento
an ebay inc company

SUPEE-6788 and Backward Compatibility

Magento Enterprise 1.14.2.2, Community 1.9.2.2 and the patch bundle **SUPEE-6788** address several security issues. Unfortunately, addressing these issues required some changes that may possibly break backward compatibility with customizations or extensions. Below you will find a list of changes and potential issues that may arise:

APPSEC-1034, addressing bypassing custom admin URL

- If a module has admin functionality that is not under the admin URL, it will need to be modified (eg. http://domain.com/cool_module instead of http://domain.com/admin/cool_module)
- Developers need to change etc/config.xml and all code lines where they generate links to the admin part of the module.
- For example, the following config.xml file for a module:

```
<admin>
    <routers>
        <custom_module>
            <use>admin</use>
            <args>
                <module>custom_module</module>
                <frontName>custom_module</frontName>
            </args>
        </custom_module>
    </routers>
</admin>
```

- Should be changed to:

```
<admin>
    <routers>
        <adminhtml>
            <args>
                <modules>
                    <custom_module
after="Mage_Adminhtml">CustomModule_Adminhtml</custom_module>
                </modules>
            </args>
        </adminhtml>
    </routers>
</admin>
```

Magento
an ebay inc company

APPSEC-1063, addressing possible SQL injection

- Modules that use SQL statements as field names or escape the field manually will need to be modified. Examples of code that is no longer allowed:

```
$collection->addFieldToFilter('(field1 - field2)', array('eq' => 3))
$collection->addFieldToFilter('`field`', array('eq' => 3))
```

- Developers will need to change the way they generate filters for collections.
- The following code:

```
$collection->addFieldToFilter('`field`', array('eq' => 3));
```

- Should be changed to:

```
$collection->addFieldToFilter('field', array('eq' => 3));
```

- The following code:

```
$collection->addFieldToFilter('(field1 - field2)', array('eq' => 3))
```

- Should be changed to:

```
$expression = '(field1 - field2)';
$condition = $this->_getConditionSql($expression, array('eq' => 3));
$this->_select->where($condition)
```

- The following approach could be used alternatively:

```
Class T extends Mage_Core_Model_Resource_Db_Collection_Abstract {
…

protected $_map = array('fields' => array(
    'condition' => '(field1 - field2)',
);
…

public function someMethod() {
    $this->addFieldToFilter('condition', array('eq' => 3));
}
…
}
```

APPSEC-1057, template processing method allows access to private information

- Magento now includes a white list of allowed blocks or directives. If a module or anyone uses variables like `{{config path="web/unsecure/base_url"}}` and `{{block type=rss/order_new}}` in CMS pages or emails, and the directives are not on this list, you will need to add them with your database installation script. Extensions or custom code that handles content (like blog extensions) might be affected.

- A full list of allowed variables and blocks in the default installation is:

  Variables:

```
web/unsecure/base_url
web/secure/base_url
trans_email/ident_general/name
trans_email/ident_sales/name
trans_email/ident_sales/email
trans_email/ident_custom1/name
trans_email/ident_custom1/email
trans_email/ident_custom2/name
trans_email/ident_custom2/email
general/store_information/name
general/store_information/phone
general/store_information/address
```

  Blocks:

```
core/template
catalog/product_new
```

- If your code uses some config variables or blocks, you need to create a data update script that adds variables or blocks to the white list tables:

```
'permission_variable'
'permission_block'
```

APPSEC-1079, addressing potential exploit with custom option file type

- This change will affect any customization that uses product custom options to save information as a PHP object. Such an approach will no longer be possible.