Magento Security and Vulnerabilities

eltrino

Roman Stepanov

http://ice.eltrino.com/

Table of contents

The Open Web Application Security Project

https://www.owasp.org/

- The Open Web Application Security Project (OWASP) is a worldwide not-for-profit charitable organization focused on improving the security of software. It's mission is to make software security visible, so that individuals and organizations worldwide can make informed decisions about true software security risks.

- Everyone is free to participate in OWASP and all of our materials are available under a free and open software license

OWASP TOP 10 List 2013

https://www.owasp.org/

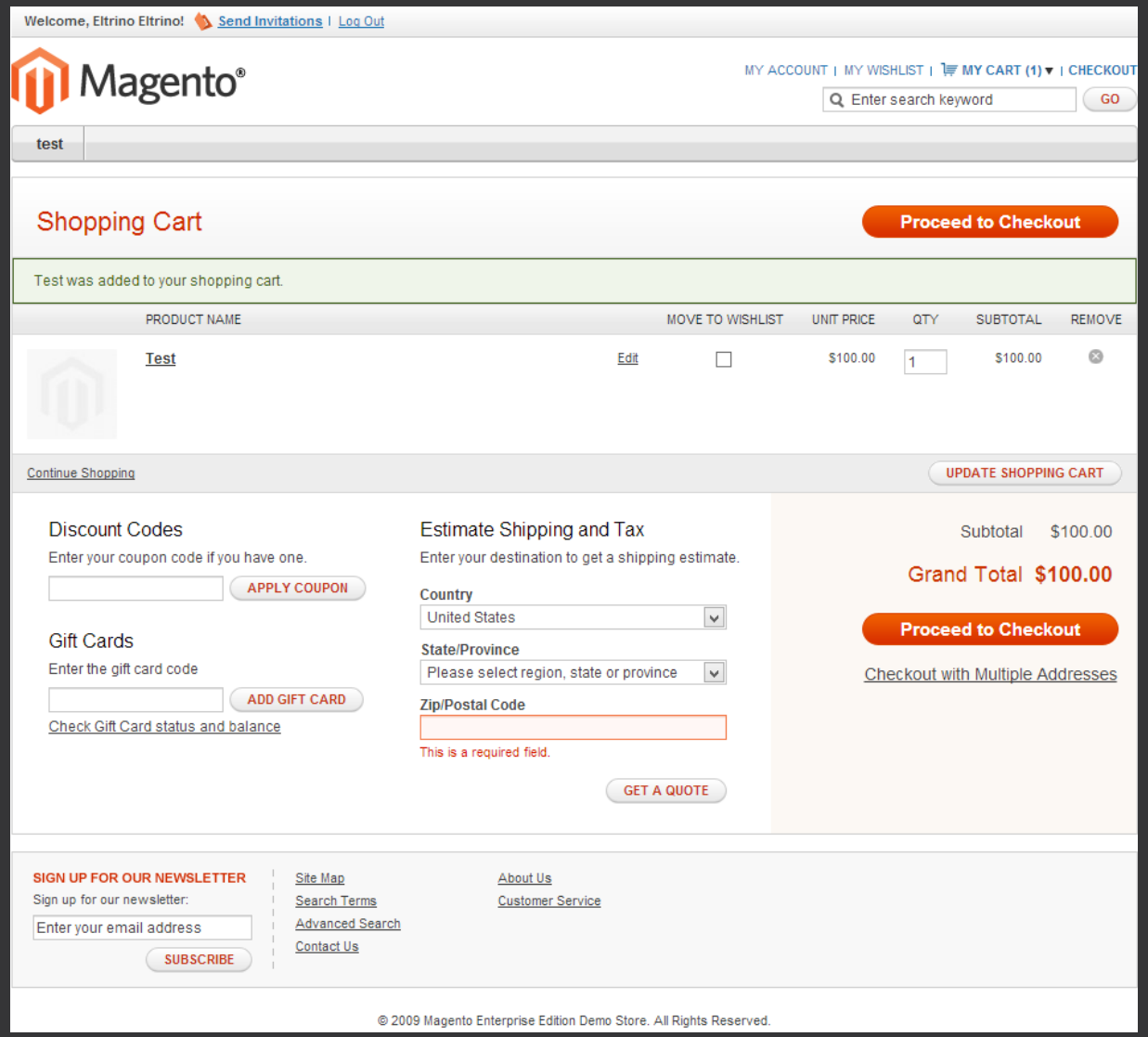
The Open Web Application Security Project

A1 – Injection
A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References
A5 – Security Misconfiguration
A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control
A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards

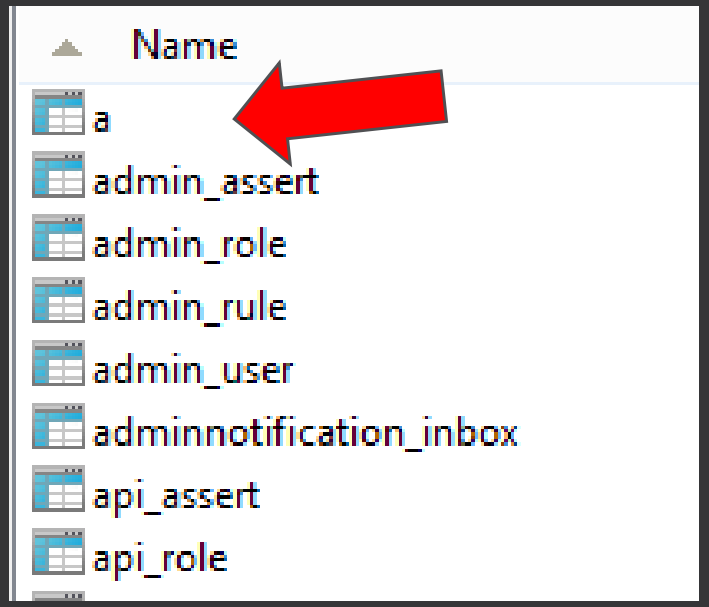
A1 – Injection

Injection flaws, such as SQL, OS injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.

A1 – Injection



10)); CREATE TABLE a (id INT); - 100



A1 – Injection: How to…

Black Box

- Check every customer using vulnerable patterns:
  - ‘ (single quote)
  - “ (double quote)
  - ` (back quote)
  - ; (semicolon)
- Enable system and exception logging
- Enable advanced sql logging
- Do not forget! Different logic for different kinds of provided data

White Box

- Code review
  - Search patterns
- Dedicated “A-Team”
  - Information and education inside company
  - Code review of new functionality
- Communication

A1 – Injection: How to…


```
105
106         /**
107          * Safely quotes a value for an SQL statement.
108          *
109          * If an array is passed as the value, the array values are quoted
110          * and then returned as a comma-separated string.
111          *
112          * @param mixed $value The value to quote.
113          * @param mixed $type  OPTIONAL the SQL datatype name, or constant, or null.
114          *                     Zend_Db::INT_TYPE, Zend_Db::BIGINT_TYPE и Zend_Db::FLOAT_TYPE
115          * @return mixed An SQL-safe quoted value (or string of separated values).
116          */
117         Zend_Db_Adapter_Abstract::quote($value, $type = null);
118
119
```

A1 – Injection: How to…

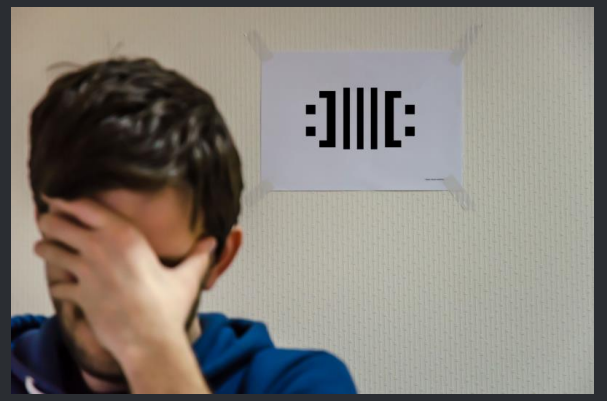

```
105
106         /**
107          * Quotes a value and places into a piece of text at a placeholder.
108          *
109          * The placeholder is a question-mark; all placeholders will be replaced
110          * with the quoted value.
111          *
112          * @param string  $text  The text with a placeholder.
113          * @param mixed   $value The value to quote.
114          * @param string  $type  OPTIONAL SQL datatype
115          * @param integer $count OPTIONAL count of placeholders to replace
116          * @return string An SQL-safe quoted value placed into the original text.
117          */
118         Zend_Db_Adapter_Abstract::quoteInto($text, $value, $type = null, $count = null);
119
```

A1 – Injection: How to…


```
94
95
96          Zend_Db_Select::where($condition, $value = null, $type = null);
97          Zend_Db_Select::orWhere($condition, $value = null, $type = null);
98          Zend_Db_Select::having($condition, $value = null, $type = null);
99          Zend_Db_Select::orHaving($condition, $value = null, $type = null);
100
101
102         Varien_Db_Adapter_Interface::prepareSqlCondition($fieldName, $condition);
103         Varien_Data_Collection_Db::addFieldToFilter($field, $condition = null);
104         Mage_Eav_Model_Entity_Collection_Abstract::addAttributeToFilter($attribute,
105             $condition = null, $joinType = 'inner');
106         Mage_Eav_Model_Entity_Collection_Abstract::addFieldToFilter($attribute,
107             $condition = null);
108
```

A1 – Injection: How to…

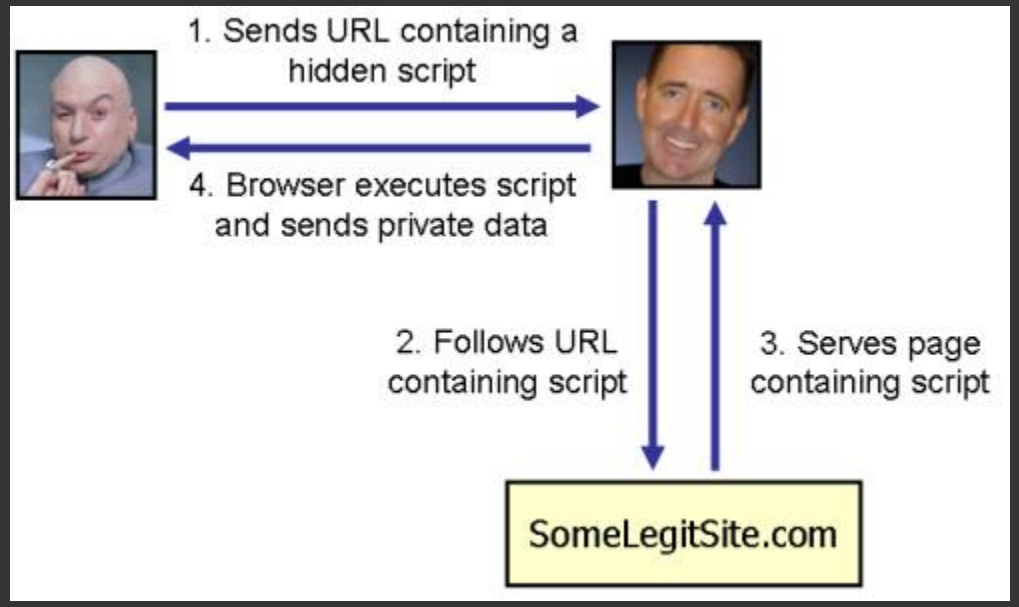

```
416         $someValue_a = 'params1';
417         $someValue_b = 'params2';
418         $select = $this->getSelect();
419         $adapter = $select->getAdapter();
420
421         /* Good */
422         $condition = $adapter->quoteInto('table.field = ?', $someValue_a);
423
424         /* Bad */
425         $condition = 'table.field = ' . $someValue_a;
426         $condition = $adapter->quoteInto('table.field_a = ? && table.field_b = ' . $someValue_a, $someValue_b);
427
```



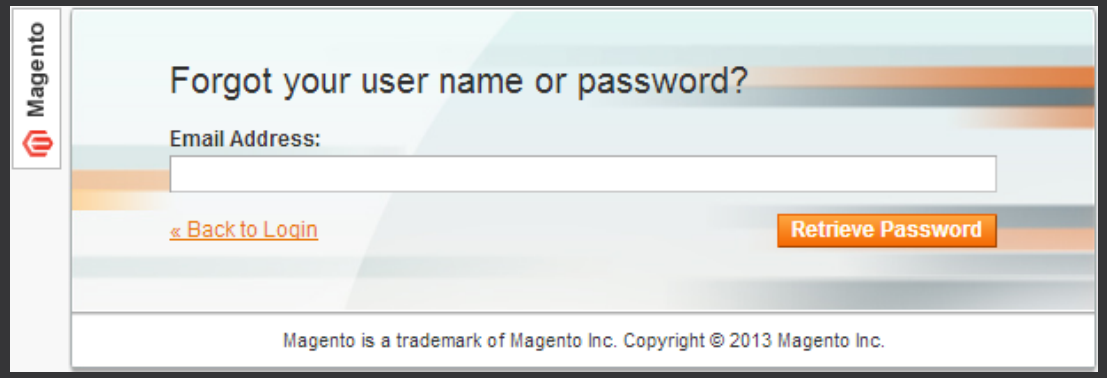
Remember about type hinting!

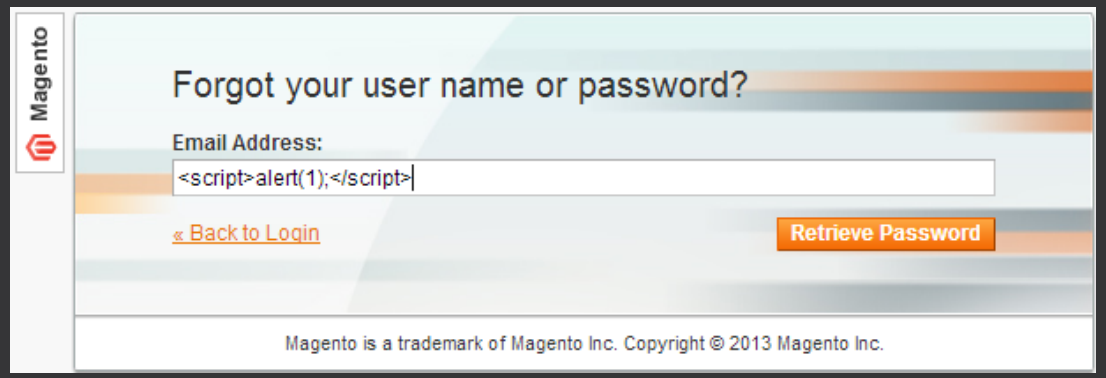
A3 – Cross-Site Scripting (XSS)

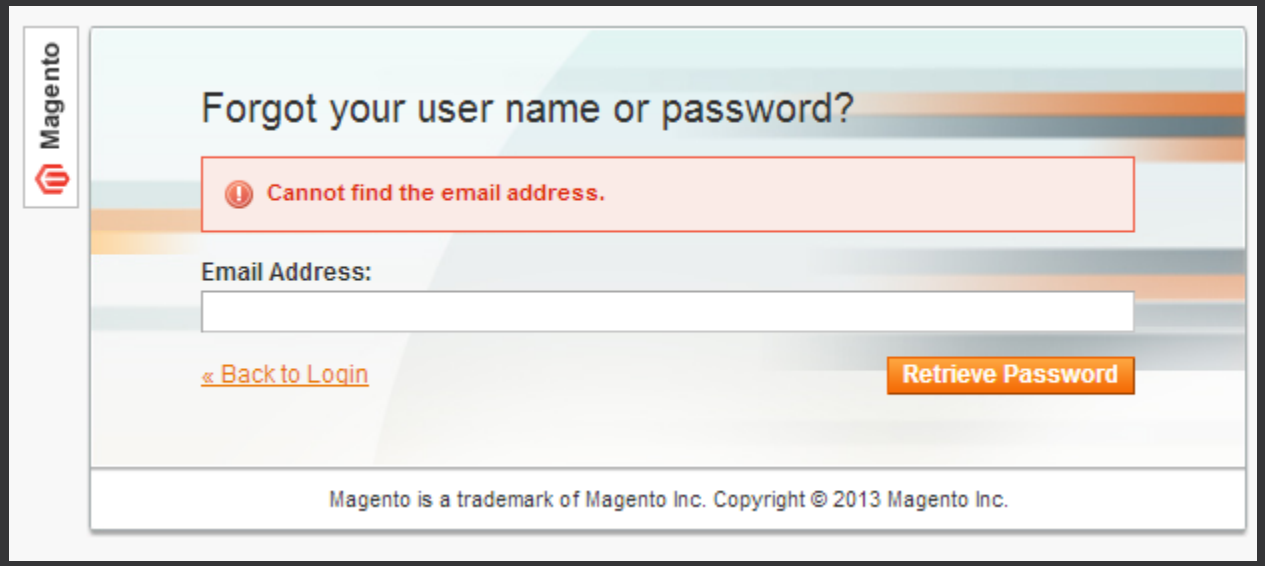
XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.



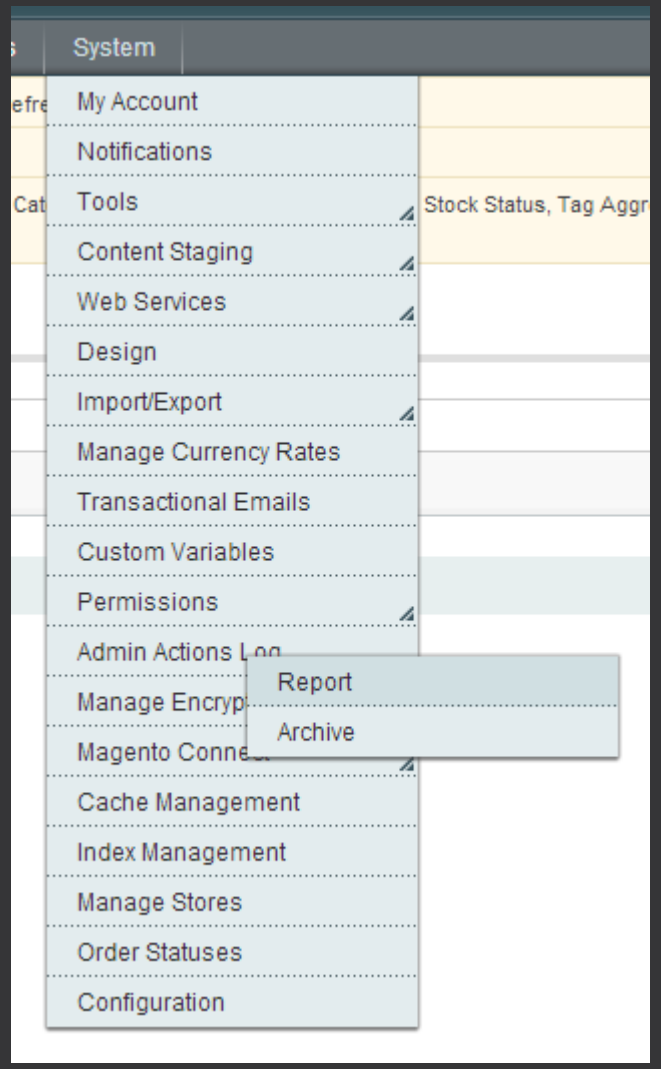
A3 – Cross-Site Scripting (XSS)

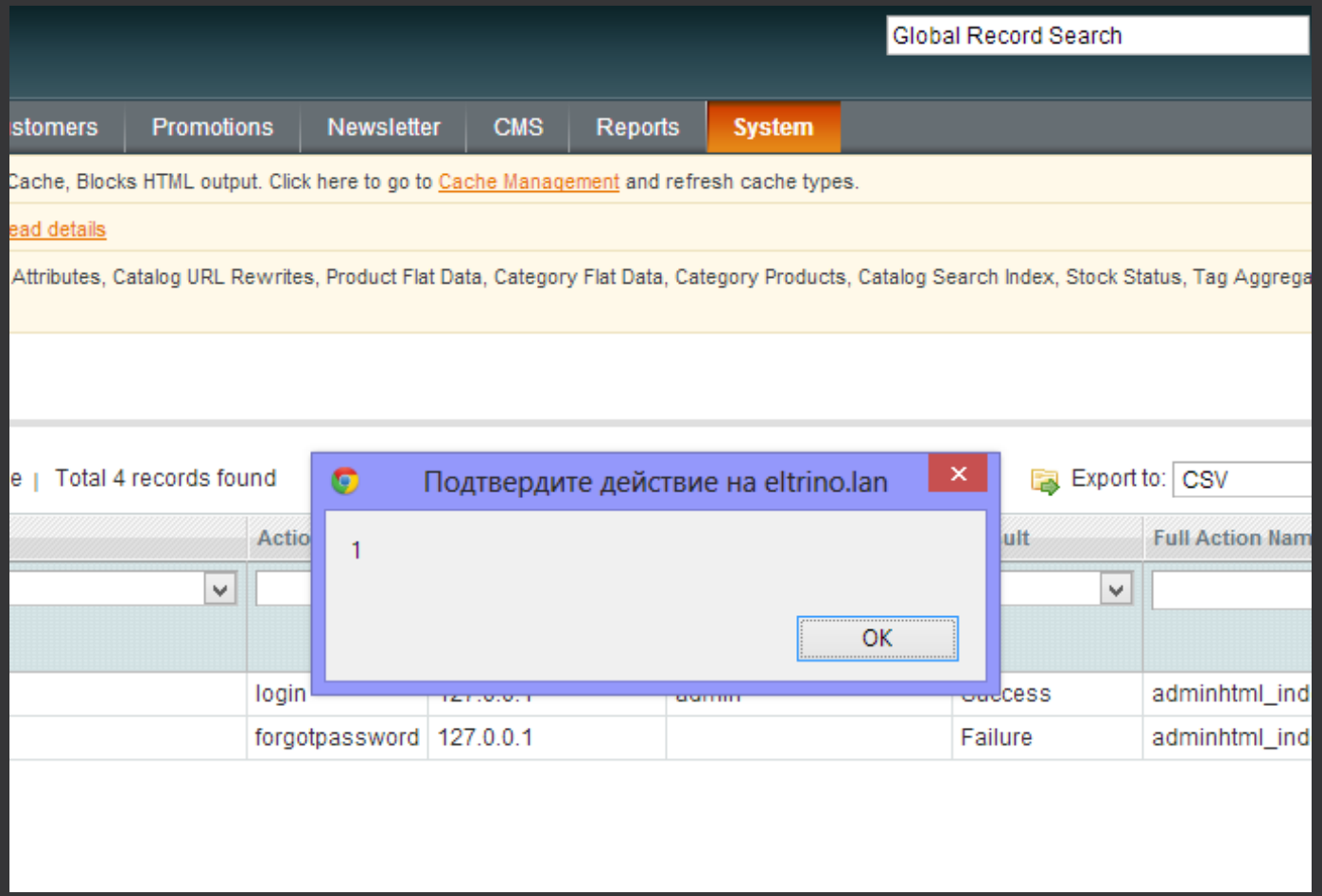






A3 – Cross-Site Scripting (XSS)





A3 – Cross-Site Scripting (XSS): How to…

- If it is possible, sanitize data on input;
- Always escape special characters on output in templates;
- Use WhiteBox testing. All variables used in input forms should be analyzed;
- Remember, that stored XSS does not need a malicious link to be exploited. A successful exploitation occurs when a user visits a page with a stored XSS.
- Quotation (', ") or other ways to escape characters isn't enough because of <script>alert(String.fromCharCode(88,83,83))</script>

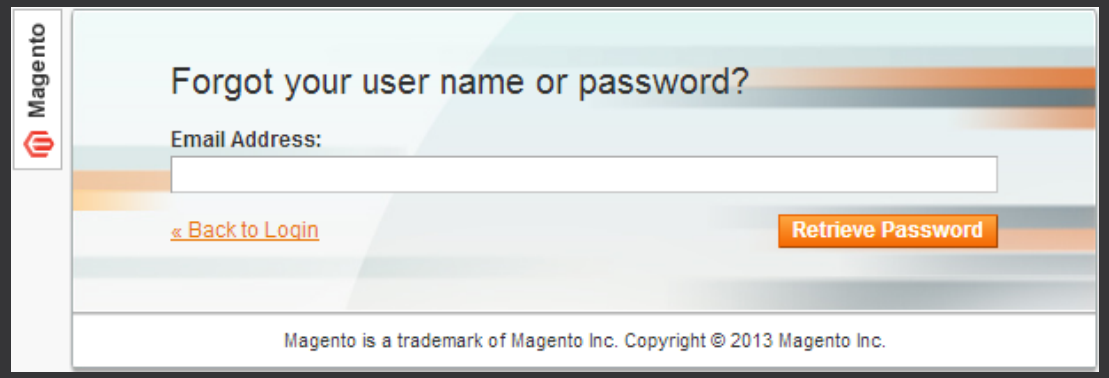
A3 – Cross-Site Scripting (XSS): How to…

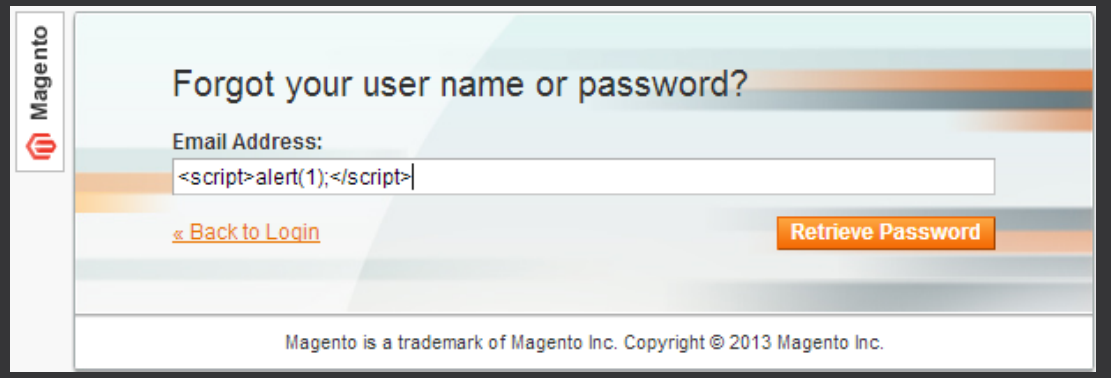

```
418
419     /**
420      * Escape html entities
421      */
422     Mage_Core_Helper_Abstract::escapeHtml($data, $allowedTags = null);
423     Mage_Core_Block_Abstract::escapeHtml($data, $allowedTags = null);
424
425     /**
426      * Wrapper for standard strip_tags() function with extra functionality for html entities
427      */
428     Mage_Core_Helper_Abstract::stripTags($data, $allowableTags = null, $allowHtmlEntities = false);
429     Mage_Core_Block_Abstract::stripTags($data, $allowableTags = null, $allowHtmlEntities = false);
430
431
432
433     /**
434      * Filter for removing malicious code from HTML
435      */
436     Mage_Core_Model_Input_Filter_MaliciousCode::filter($value);
```

A8 – Cross-Site Request Forgery (CSRF)

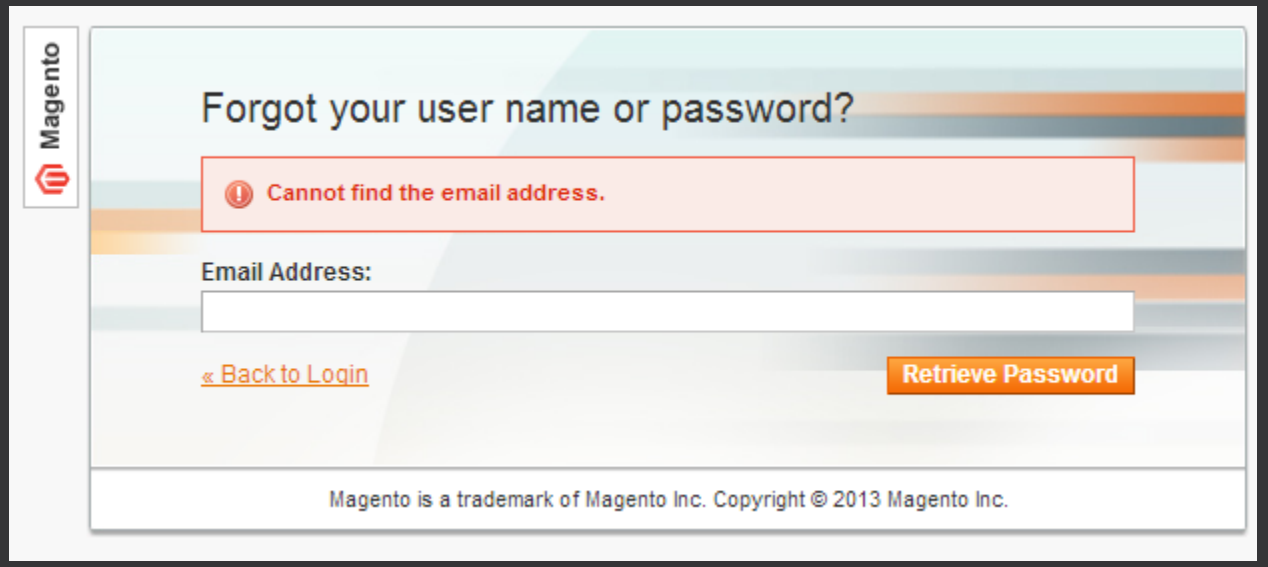
A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

A8 – Cross-Site Request Forgery (CSRF)

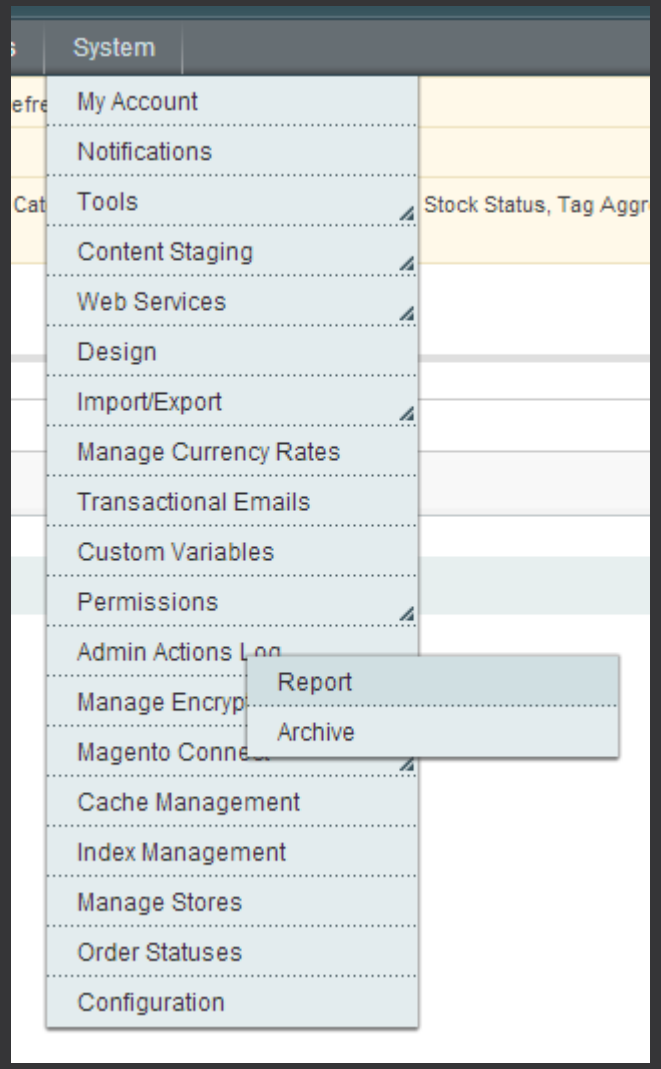


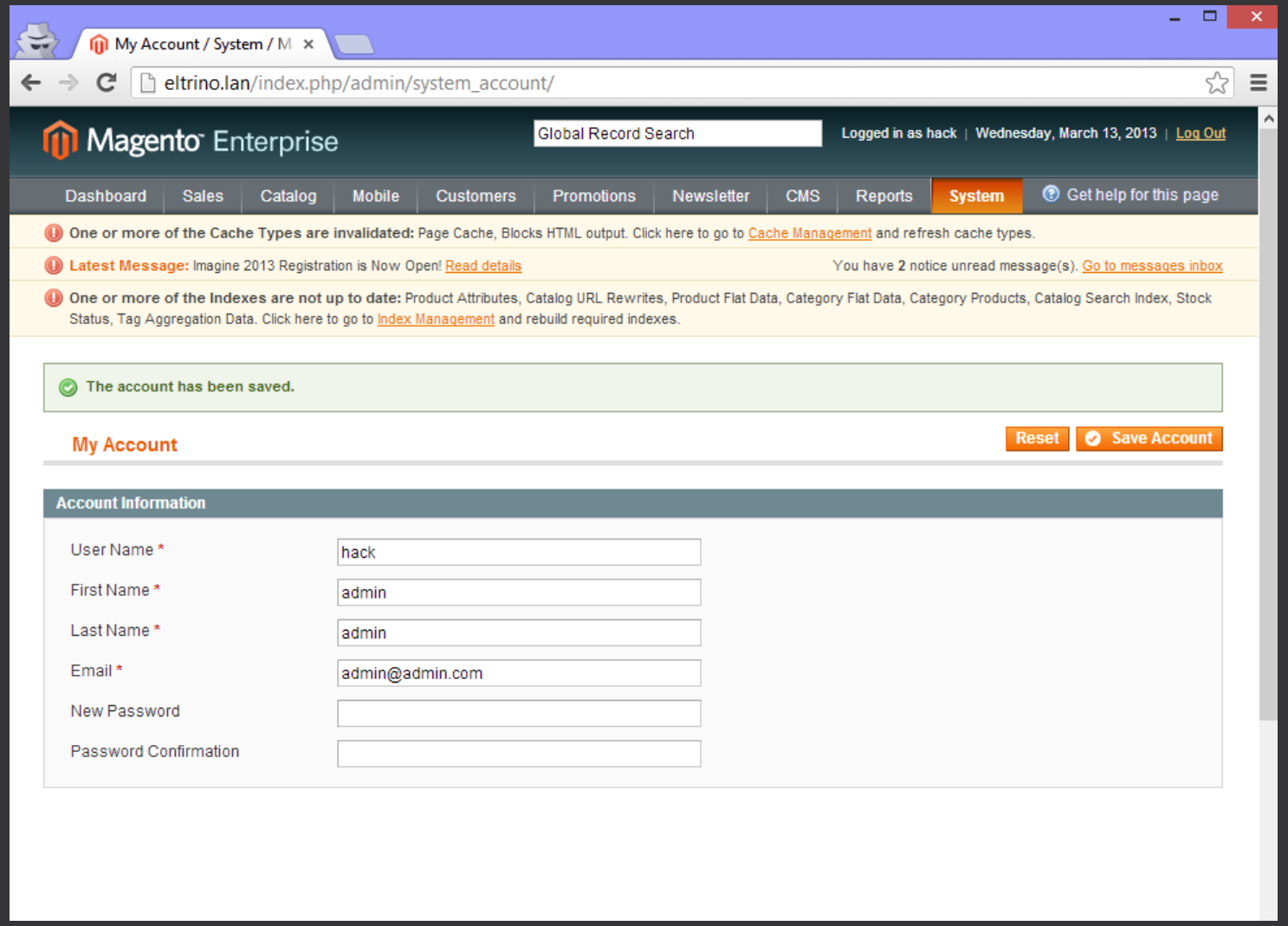


document.location.href="http://eltrino.lan/index.php/admin/system_account/save/?username=hack&firstname=admin&lastname=admin&email=admin@admin.com&new_pasword=hack&password_confirmation=hack"



A8 – Cross-Site Request Forgery (CSRF)





A8 – Cross-Site Request Forgery (CSRF)

- Verify that secure hash is used for URLs
- Verify that HTML forms have form_key attribute, and form cannot be submitted with incorrect form_key

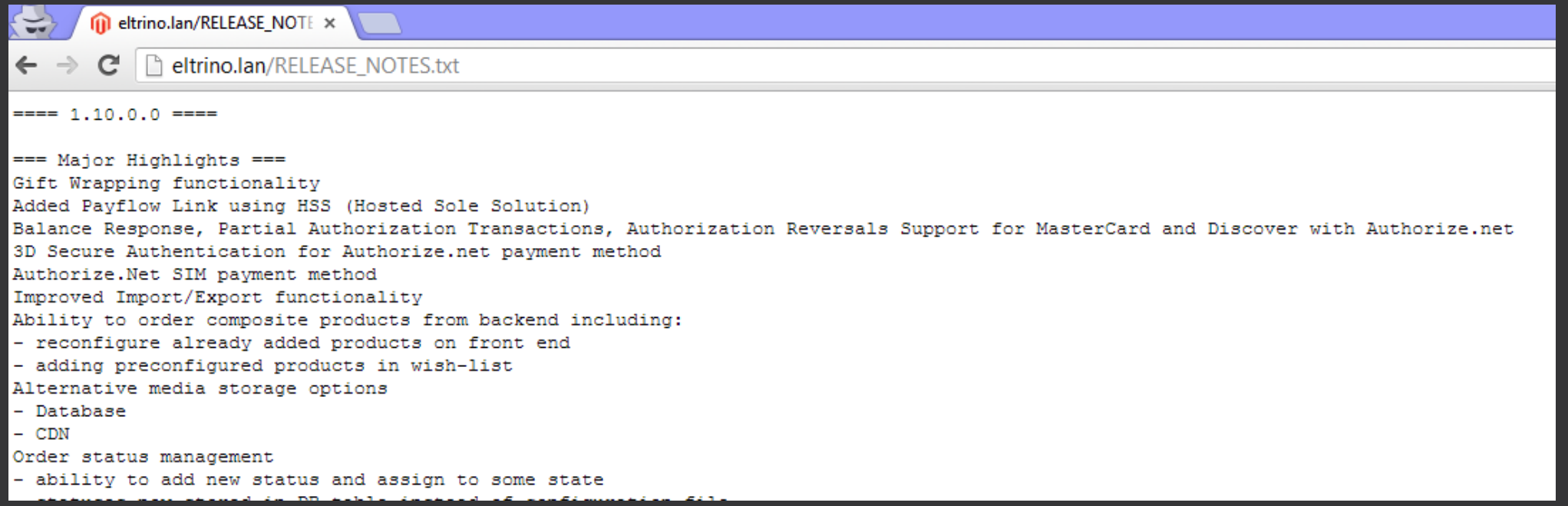
A8 – Cross-Site Request Forgery (CSRF)


```
118        /**
119         * Retrieve Session Form Key
120         *
121         * @return string A 16 bit unique key for forms
122         */
123        Mage_Core_Model_Session::getFormKey();
124
125        /**
126         * Adds form_key field to form
127         */
128        Varien_Data_Form::toHtml();
129
130        /**
131         * Validates for secret key if request was submitted via Post
132         */
133        Mage_Core_Controller_Varien_Action::preDispatch();
134
135        /**
136         * Generate secret key for contreoller and action based on form key
137         */
138        Mage_Adminhtml_Model_Url::getSecretKey($controller = null, $action = null);
139
140        /**
141         * Adds secure key to URL if needed
142         */
143        Mage_Adminhtml_Model_Url::getUrl($routePath=null, $routeParams=null);
```

A5 – Security Misconfiguration

Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. All these settings should be defined, implemented, and maintained as many system are shipped with defaults that are not secure enough. This also means keeping all software up to date on regular basis.

A5 – Security Misconfiguration



A5 – Security Misconfiguration: How to…

- Consider the threats you plan to protect this data from (e.g., insider attack, external user), make sure you encrypt all sensitive data at rest and in transit in a manner that defends against these threats.
- Don’t store sensitive data unnecessarily. Discard it as soon as possible. Data you don’t have can’t be stolen.
- Ensure strong standard algorithms and strong keys are used, and proper key management is in place.
- Disable autocomplete on forms collecting sensitive data and disable caching for pages displaying sensitive data.

3rd party integrations

Most of eCommerce solutions required integration with 3rd part systems like

- Payment methods
- Shipping methods
- ERP systems

In result of incorrect implementation

- Sensitive information about your client or his customers could be exposed to attacker.
- Attacker can change workflow in system for his own needs (order status, payments etc)

3rd party integrations

How to prevent it?

- If 3rd party system allows usage of some secret keys/words/hashes/…/etc always perform validation of these parameters for each request
- Don’t place export files with sensitive information into folders accessible through the web
- Don’t use encryption/decryption of data using Magento Secret Key which can be posted outside
- If it’s possible accept connections only from certain ip range

QA

Roman Stepanov
roman@eltrino.com
skype: mail2punk

Q&A