



ADITYA K. SOOD

# Hacking RSS Feeds: Insecurities in Implementing RSS Feeds

Difficulty



This paper sheds light on the insecure coding practices that affect RSS based web applications and also on their flexibility. The advent of Web 2.0 has enhanced the mobility of content. The inclusion of content has become the sole basis for the inter-working of websites.

**R**SS feeds are used extensively. This serves as an interdependent working platform. But during penetration testing sessions, PHP based RSS applications show vulnerable behavior due to insecure coding. As a result of this, web application robustness is affected. This layout is versatile from a security point of view as well as from a working structure of applications. This paper discusses the infection vectors that occur due to insecure coding by developers and includes other related security issues. It will provide a detailed analysis of the errors and efficient measures to correct those errors, while keeping in mind the original security concerns.

- Guidelines should be provided which are to be followed by the scraper.
- Simply block the IP Address at the interface level.
- Develop Feeds manually with constraints.

## Working of an RSS Enabled Application

The working paradigm starts from a well designed Content Management System. It is considered to be the root of feed transference and the operations required in managing data flow between various websites. The generation and transference of feeds is based on the application coding used for web services. PHP is extensively used for creating the feeds structure. The model shows the inclusion of feeds from different websites and their processing by the service programs. Now let's look at the general structure of Content Management and Syndication: see Figure 1.

The user is one of the components of this management system because the major interaction is undertaken with the user. The content management system always produces HTML and RSS Feeds. The language used is XML which is based on a standard specification. More precisely XSLT is used for transformation of content into feeds. The feeds can be directly converted into HTML pages based on the designed application. As a result of this operation, a direct interface is provided to a user.

## Cyber Law Perspective on this Issue

RSS Feeds follow the Copyright procedure and the implications covered in it. The major point of discussion is Site Scraping. It is a process of scraping the contents of another website into a different format. The US laws have rightly stated that linking to another website is not a breach of the Copyright Act. It becomes an issue whenever the site owner wants you to stop scraping and it is still continued in an illegal way. The best approach to follow is Licensed Feeds.

The best practices are:

- The scraper should follow the robots.txt directives.

### WHAT YOU WILL LEARN...

Peripheral knowledge of working of RSS feeds will be useful.

Developers must know RSS implementation in PHP and ASP

### WHAT YOU SHOULD KNOW...

The insecure elements in RSS implementation which results in Web Exploitation

Developmental problems in implementing RSS

Security impacts due to RSS Flaws

The RSS configurations provide large quantities of information as feed elements to remote sites for de-centralizing information. There are a number of RSS variants present with different specifications. The base structure works on the specification used as a benchmark for designing elemental objects. The prime mechanism is the same for any kind of RSS implementation.

To understand security implications it is crucial to comprehend the parsing of RSS feeds. The feeds are parsed through three basic techniques which are enumerated below.

### XML Parsing

This is a process of parsing the raw feeds into well structured RSS feeds to be used directly in website content and blog feeds. For detailed lookup, let's see a short PERL code for practical implementation of XML parsers. It is implemented through the XML::Simple parsing library. See Listing 1.

### Implementing Regular Expressions

Regular expressions can be used effectively for parsing RSS feeds. To do this requires no module installation and can be directly applied throughout the program. No doubt the implementing of regular expression is a somewhat complex procedure. An unstructured regular expression can affect the stability of an application. Let's have a look at this code snippet: see Listing 2.

### XML/XSLT Transformations

The transformation is basically done to generate style sheets that are used as such for RSS feeds. It's basically a reproduction process for arranging metadata by reducing the complexities in relationship between the different objects used. Let's look at the transformation mechanism between XML/XSLT: see Figure 2.

These components are well placed in the above presented hierarchical model of RSS generation. Let's look at a very simple RSS field with one item: see Listing 3.

Basically the feeds structure is based on the following:

- Channel (title, description, URL, creation date, etc.)

#### Listing 1. Simple Parsing Library

```
use LWP::Simple;
use XML::Simple;

my $url=$ARGV[0];

# Retrieve the feed, or die gracefully
my $feed_to_parse = get ($url) or die "I can't get the feed you want";

# Parse the XML
my $parser = XML::Simple->new( );
my $rss = $parser->XMLin("$feed_to_parse");

# Decide on name for outputfile
my $outputfile = "$rss->{'channel'}->{'title'}.html";

# Replace any spaces within the title with an underscore
$outputfile =~ s/ /_/g;

# Open the output file
open (OUTPUTFILE, ">$outputfile");

# Print the Channel Title
print OUTPUTFILE '<div class="channelLink">'. "\n". '<a href="';
print OUTPUTFILE "$rss->{'channel'}->{'link'}. ' ">';
print OUTPUTFILE "$rss->{'channel'}->{'title'}</a>\n</div>\n";

# Print the channel items
print OUTPUTFILE '<div class="linkentries">'. "\n". "<ul>";
print OUTPUTFILE "\n";

foreach my $item (@{$rss->{'channel'}->{'item'}}) {
    next unless defined($item->{'title'}) && defined($item->{'link'});
    print OUTPUTFILE '<li><a href="';
    print OUTPUTFILE "$item->{'link'}";
    print OUTPUTFILE "' ">';
    print OUTPUTFILE "$item->{'title'}</a></li>\n";
}

foreach my $item (@{$rss->{'item'}}) {
    next unless defined($item->{'title'}) && defined($item->{'link'});
    print OUTPUTFILE '<li><a href="';
    print OUTPUTFILE "$item->{'link'}";
    print OUTPUTFILE "' ">';
    print OUTPUTFILE "$item->{'title'}</a></li>\n";
}

print OUTPUTFILE "</ul>\n</div>\n";

# Close the OUTPUTFILE
close (OUTPUTFILE);
```

#### Listing 2. Code Snipped

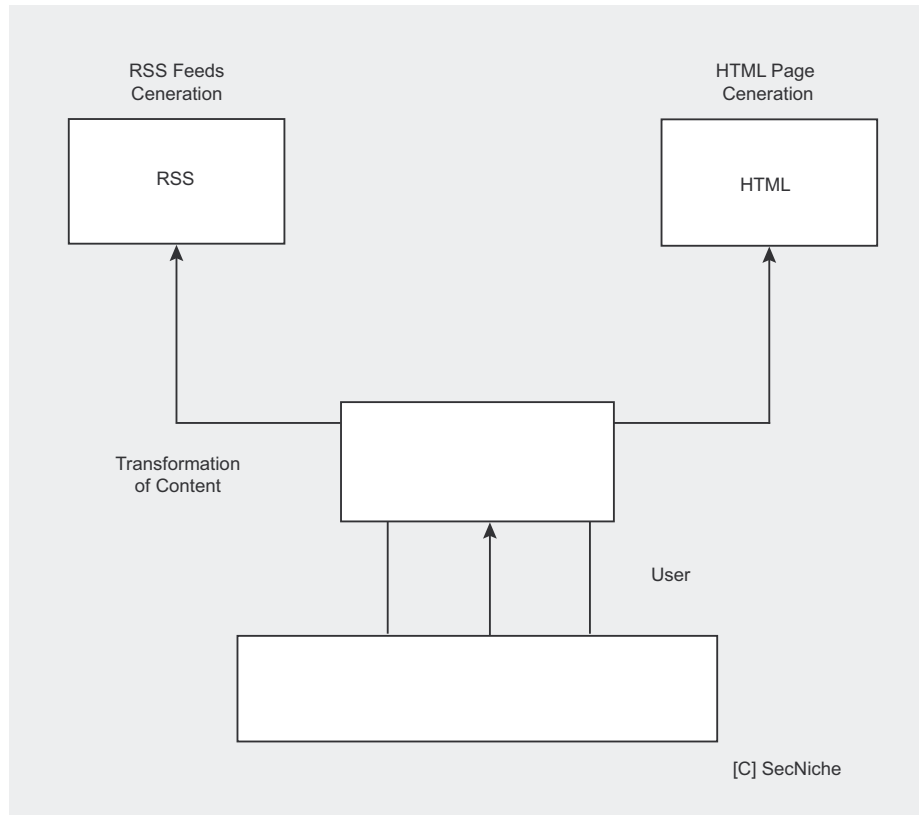
```
# Feed's title and link
my($f_title, $f_link) = ($rss =~ m{<title>(.*?)</title>.*?<link>(.*?)</link>#ms});

# RSS items' title, link, and description

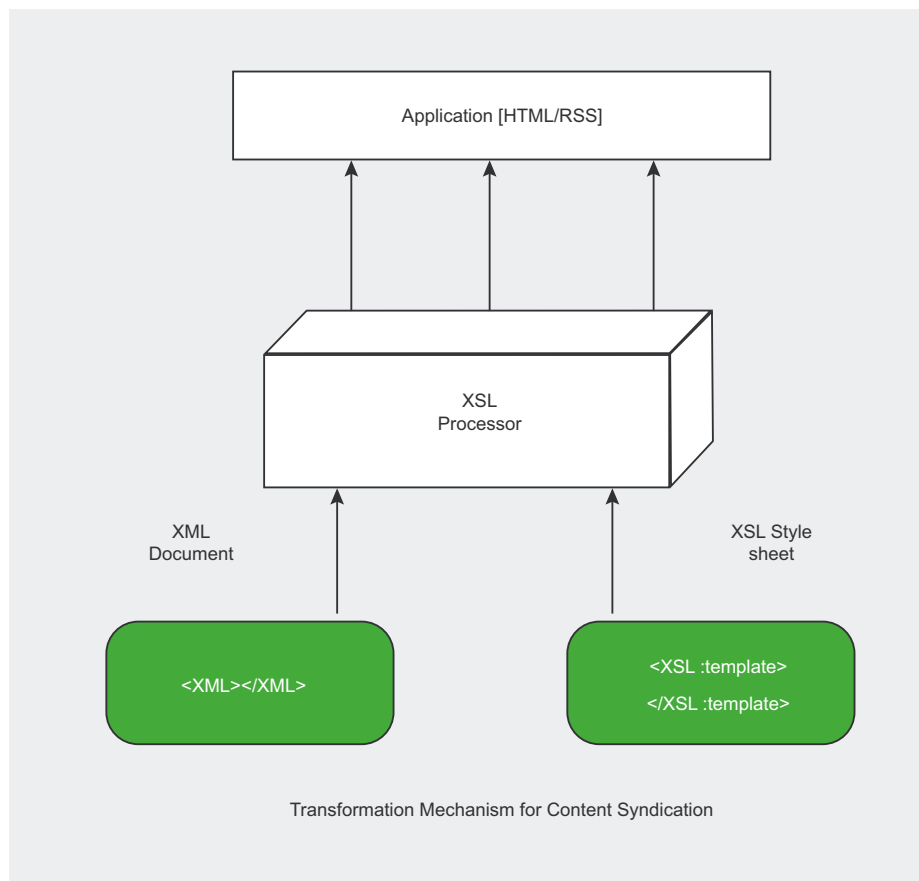
while ( $rss =~ m{<item(?:s)?.*?<title>(.*?)</title>.*?<link>(.*?)</link>.*?<description>(.*?)</description>.*?</item>}mgis ) {
    my($i_title, $i_link, $i_desc, $i_fn) = ($1||'', $2||'', $3||'', undef);

    # Unescape &amp; < > to produce useful HTML
    my %unescape = ('<'=>'<', '>'=>'>', '&'=>'&', '"'=>'");
    my $unescape_re = join '|', => keys %unescape;
    $i_title && $i_title =~ s/($unescape_re)/$unescape{$1}/g;
    $i_desc && $i_desc =~ s/($unescape_re)/$unescape{$1}/g;
```

- Image
- Item (title, description, URL, etc.)
- Item (title, description, URL, etc.)



**Figure 1.** RSS Working Layout



**Figure 2.** RSS Content Syndication – Transformation

This structure is converted into a well defined object component that can then be easily placed into HTML pages or other web services for reproduction of data in an efficient manner. This discussion provides a brief overview of RSS functioning. As our discussion is geared more towards application flaws, we will now look into various insecure practices one by one.

## RSS Attack Vectors – Developmental Insecurities

Now we will discuss various attack vectors and insecure coding practices used by developers by analyzing a number of stringent errors.

### Failure in Calling DOM Based Functions

Below is one of the function-related specific errors in the RSS based applications. The call to `domxml_new_doc()` fails as a result of which an application is unable to create a new document object. The function `domxml_new_doc()` is a modified version of `domxml_new_xmldoc()`.

```
Fatal error: Call to undefined function
domxml_new_doc() in /home/.bauzeur/
thecabin/podcast/rss.php on line 41
```

This specific function is mainly used in PHP5. There are a number of problems that arise from migration from PHP4 to PHP5. The applications' functionality depends on the software versions too. The function creates a new empty XML document and returns an instance of it. The XML number version of the document is passed as an argument. The function prototype is structured as:

```
DomDocument domxml_new_doc (string
$version)
```

The XML documents are used to transfer data in XML format which are further rendered by the browser for better presentation to the user. The background working of RSS is based on the generation and handling of XML documents that possess data. The function is called from the GNOME XML library. The calling mechanism uses

php\_xmldom.dll. The DLL provides dynamic loading of a number of functions defined in it. It is configured by specifying it as an entry in the php.ini file. The function prototype should be defined as: see Listing 4.

The code is symmetrical and is used as a definitive code structure. The developers should focus on the calling method and the migration of content between different web pages. The kind of PHP version to be used with XML document creation also impacts the robustness of the web application. Of course secure coding is very important.

It affects the security of an application too. Wrong definitions of DTD's can be a problem because a badly crafted or malicious XML document makes the

parser consume the CPU time and memory extensively thereby resulting in a potential Denial of service. Many developers or system controllers enable DTD which is considered to be a security risk. Even base software disables DTD's by default. Another point that comes into play is whether to accept DTD from other resources or not. For security, the sources have to be trusted. Any DTD from an untrusted source generates vulnerable behavior. The XPATH problems can be encountered if the code is designed to call remote objects from an untrusted source. The developers should concentrate on this factor because, again, it results in potential denial of service with the inclusion of complex queries by the malicious user.

### Listing 3. Code Snippet for Transforming into XHTML Fragments

```
<rss version="2.0" xmlns:dc="http://purl.org/dc/elements/1.1/">
<hacker>
<title>RSS2.0 Flaws</title>
<link>http://www.exampleurl.com/example/index.html</link>
<description>This is an example RSS 2.0 feed Flaws</description>
<language>en-gb</language>
</hacker>
</rss>
```

Example: Code Snippet for transforming into XHTML fragments

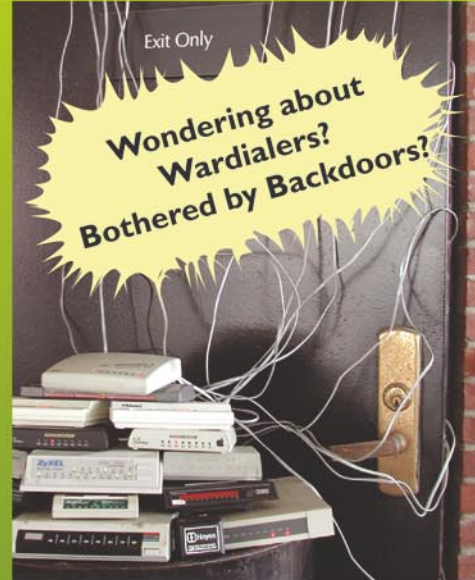
```
<?xml version="1.0"?>

<xsl:stylesheet version = '1.0'
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rss="http://purl.org/rss/1.0/"
exclude-result-prefixes="rss rdf"
>
<xsl:output method="html"/>

<xsl:template match="/">
<div class="channellink">
<a href="{rdf:RDF/rss:channel/rss:link}">
<xsl:value-of select="rdf:RDF/rss:channel/rss:title"/>
</a>
</div>
<div class="linkentries">
<ul>
<xsl:apply-templates select="rdf:RDF/*"/>
</ul>
</div>
</xsl:template>

<xsl:template match="rss:channel|rss:item">
<li>
<a href="{rss:link}">
<xsl:value-of select="rss:title"/>
</a>
</li>
</xsl:template>

</xsl:stylesheet>
```



**PhoneSweep®**  
gives you the answers

Get PhoneSweep, the original audit-quality multi-line telephone scanner.

- Dial using 1-16 lines
- Patented Single Call Detect technology
- Detects carrier, fax, voice, busy, second dial tone, timeout, untrained carrier
- Identifies over 465 systems
- Support for Microsoft Windows 98 through VISTA
- Since 1998

New! Just released  
PhoneSweep 5.5 provides:

- ★ Adds Support for Microsoft's Windows VISTA
- ★ Email notification and optional Gold remote access supports IPv6
- ★ Continuous Scan checks your remotely-accessible servers, fax lines or ACD lines and reports availability changes in real time
- ★ New user-selectable reporting categories



**Sandstorm Enterprises®**  
tools with sharp edges®

www.sandstorm.net  
+1 781.333.3200 • sales@sandstorm.net

## Listing 4. Function Prototype

```

define('DOMXML_LOAD_PARSING',0);
define('DOMXML_LOAD_VALIDATING',1);
define('DOMXML_LOAD_RECOVERING',2);
define('DOMXML_LOAD_SUBSTITUTE_ENTITIES',4);
define('DOMXML_LOAD_DONT_KEEP_BLANKS',16);

[PHP5] - function domxml_new_doc($version) {return new php4DOMDocument();}
[PHP4] - function domxml_new_xmldoc($version) {return new php4DOMDocument();}

```

So the code would be stated as:

```

$dom = new DOMDocument('1.0');

// create and append the root element, <rss>
$rss = $dom->appendChild($dom->createElement('rss'));

// create and append <error_check> to $rss
$error_check = $rss->appendChild($dom->createElement('error_check'));

// set the text node for $error_check
$error_check->appendChild($dom->createTextNode('PHP DONE'));

// print DOM document as XML
echo $dom->saveXML();

```

The other way can be:

```

$dom = new DOMDocument('1.0');
// create and append the root element, <rss>
$rss = $dom->appendChild ($dom-> createElement ('rss'));

// create and append <title> to $rss
$rss->appendChild ($dom-> createElement ('title'));

// set the text node for $title
$error_check ->appendChild ($dom-> createTextNode ('PHP DONE!'));

// print DOM document as XML
$dom->formatOutput = true;
echo $dom->saveXML();

```

## Listing 5. Vulnerable Application Triggering a Script

```

<div id="header">
  <div id="headerimg">
    <div class="header_container">
      <h1><a href="<?php echo get_settings('home'); ?>/>"><?php bloginfo('name');
        ?></a></h1>
      <div class="description"><?php bloginfo('description'); ?></div>
    </div>
  </div>
</div>

```

## Listing 6. Structural Code

```

<?php
$rss = array("element_a", "element_b", "element_c");
reset($rss);
while (list(, $value) = each($rss))
{
echo "Value: $value<br />\n";
}

foreach ($rss as $value) {
echo "Value: $value<br />\n" }
?>

```

This in turn affects the robustness of the web application and the transference mechanism slows down.

These specific security issues can be the result of error prone XML base.

## Header Modification Checks

The rss.php web pages are prone to header based errors. The web page is divided into two parts, the header and the body. It depends a lot on the type of content to be transferred and taken in response. The testing process is termed as HTTP Response Splitting. By default the content type is set to text/html. Since the process is part of the header specification, it is crucial to modify headers based on the application requirements. The RSS based applications require XML as content type. The XML based document structure is used for data transference. Let's see:

Generally, the headers are sent as:

```

Date: Mon, 10 Jul 2007 15:51:59 GMT
Server: Apache/2.2.0 (Unix) mod_ssl/
2.2.0 OpenSSL/0.9.5g
Content-Encoding: gzip
Content-Type: text/html

```

This is a general view. For XML data, the headers have to be manipulated to a different content type as presented below:

```

Date: Mon, 10 Jul 2007 16:55:59 GMT
Server: Apache/2.2.0 (Unix) mod_ssl/
2.2.0 OpenSSL/0.9.7g
Content-Encoding: gzip
Content-Type: text/html + xml

```

This is an actual view. The developers design robust and dynamic RSS based web applications. Code writers design the code with certain standards. For example, the headers have to be specified on the basis of the output to be produced.

```

[php]
Warning: Cannot modify header
information - headers already sent by
(output started at /home.10.19/www/
blog/rss.php:2) in /home.10.19/www/
blog/rss.php on line 2

```

The PHP itself is an intelligent element and can perform certain work itself, without

any intervention by the user. The problem occurs when some of the body is sent to the user. A request to change the header is made. Since the error is a result of the header statement, the developer should look for adjacent code near the `header()` function. The basic flaw is in the use of the header function. The second cause can result from redirection of pages. The underlined code redirects the user to the destination i.e. `index.php`

```
<? header('Location: /index.php'); ?>
```

So a simple code error affects the robustness of an application. As RSS based applications require continuous functionality to update the site summary database remotely, the above defined two problems should be checked in `rss.php` (scripts in PHP to implement RSS automation, See Appendix) to avoid errors. This is considered one of the major potential risks in web application security.

The attacker can easily exploit the insecure web application by passing a simple PHP script in which headers are modified directly. For Example, a vulnerable application can trigger a script as provided Listing 5.

It changes the execution flow of the web application. In these types of attacks, the user is redirected towards the destination object which is passed as an argument to the header function. This attack basically occurs at the backend. It further acts as a base for third party redirection attacks, phishing and cross site request forgery attacks. An attacker can specify the destination URL with arguments and pass it to the web application by a simple inclusion mechanism. Once the script is injected and executed the vulnerable application is exploited according to the attacker's request. Therefore, it should be taken into account that security should be implemented through secure code designing.

## Invalid Argument Checks in Control Structures

The errors based on calling control structures are quite common. The RSS based web applications are prone to these types of errors. Usually the base is PHP

coding. Calling of structures in a wrong manner generates an error. The main problem is the passing of arguments. The basic problem which occurs with these types of errors is:

- The calling of variables not initialized in the context of code.
- The calling of variables with different data types that are not defined.

- The iteration of objects that are undertaken practically by using control structures.

The error presented below is the consequence of failure of arguments in `foreach()` control structure. The developers design an error prone code mostly while using this control structure. It basically works on arrays. When this

### Listing 7. Usage of Arrays – Example

```
function base_debug_getHash($method_name, $params, $app_data) {
    $key1 = '786'. chr(0x00);
    $key2 = '645'. chr(0x00);

    $result = array(
        $key1 => 'key1 is a '. gettype($key1),
        $key2 => 'key2 is a '. gettype($key2),
    );
    return $result;
}
xmlrpc_server_register_method($xmlrpc_server, 'base_debug.getHash',
'base_debug_getHash');
```

### Listing 8. Generating Feeds

```
XML parsing failed: syntax error (Line: 2, Character: 0)
Reparse document as HTML
Error:unexpected start-tag (root element already specified)

Specification:http://www.w3.org/TR/REC-xml/
1: <br />
2: <b>Warning</b>: filesize() [

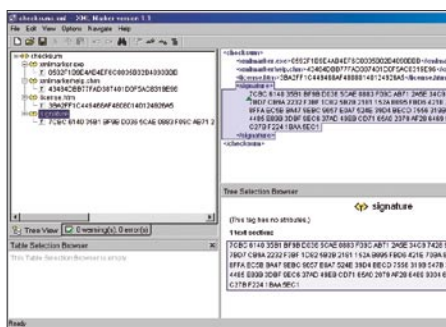
```

### Listing 9. Error Code Encountered in `rss.php`

```
Fatal error: Uncaught exception 'ADODB_Exception' with message 'mysql error: [1048:
Column 'iUserID' cannot be null] in EXECUTE("INSERT INTO
CBS_statistics (dtCreated,iUserID,iPodcastID,sCategory)
VALUES (NOW(),NULL,'32','PODCAST RSS')) ' in /home/cbsnewsr/
public_html/lib/core/External/ADODB/adodb-exceptions.inc.php:
78 Stack trace: #0 /home/news/public_html/lib/core/External/
ADODB/adodb.inc.php(886): adodb_throw('mysql', 'EXECUTE', 1048,
'Column 'iUserID...', 'INSERT INTO CBS...', false, Object(ADODB_
mysql)) #1 /home/cbsnewsr/public_html/lib/core/External/ADODB/
adodb.inc.php(842):

ADoConnection->_Execute('INSERT INTO CBS...') #2 /home/cbsnewsr/public_html/lib/core/
Objects/class.GenericObject.php(397):

ADoConnection->Execute('INSERT INTO CBS...', Array) #3 /home/cbsnewsr/public_html/
rss.php(10): GenericObject->Save() #4 {main} thrown in /
home/cbsnewsr/public_html/lib/core/External/ADODB/adodb-
exceptions.inc.php on line 78
```



**Figure 3.** XML Marker

structure is used with different objects without checking the arguments used for iteration of objects, errors occur.

Warning : Invalid argument supplied for foreach() in /var/www/gallery/rss.php pn line 126  
 Warning : Cannot modify header information - headers already sent by (output started at /var/www/gallery/rss.php 126)

### Listing 10. Error Logging Mechanism

```
<?php
$b_debugmode = 1; // 0 || 1 [Boolean Check]

$administrator_mail = 'developer@company.com';
$administrator_response_mail = 'info@mywebsite.com';

function db_query( $query ){
    global $b_debugmode;

    // Perform Query
    $result = mysql_query($query);

    // Check result
    // This shows the actual query sent to MySQL, and the error. Useful for debugging.
    if (!$result) {

        if($b_debugmode){
            $message = '<b>Invalid query:</b><br>' . mysql_error() . '<br><br>';

            $message .= '<b>Whole query:</b><br>' . $query . '<br><br>';
            die($message);
        }

        raise_error('db_query_error: ' . $message);
    }
    return $result;
}

function raise_error( $message ){
    global $administrator_mail, $administrator_response_mail;

    $serror=
        "Env:      " . $_SERVER['SERVER_NAME'] . "\r\n" .
        "timestamp: " . Date('m/d/Y H:i:s') . "\r\n" .

        "script:    " . $_SERVER['PHP_SELF'] . "\r\n" .
        "error:     " . $message . "\r\n\r\n";

    // open a log file and write error
    $fhandle = fopen( '/logs/errors'.date('Ymd').'.txt', 'a' );
    if($fhandle){
        fwrite( $fhandle, $serror );
        fclose( $fhandle );
    }

    // e-mail error to system operator
    if(!$b_debugmode)
        mail($administrator_mail, 'error: '.$message, $serror, 'From: ' . $administrator_response_mail ) ?>
```

The pointers used for array traversing in `foreach()` structures reset automatically. Let's have a look at the structural code: see Listing 6.

The `foreach()` structure is called in this manner. Developers generally prefer to use the `reset()` function. Even if this function is not used the reset is performed by the `foreach()` structure itself. One problem arises when coders do not use `unset()` function to flush the array object at the end. The main problem of error generation is caused by the arguments passed to the structure. In order to circumvent these errors, developers should define the variables carefully. The usage of arrays should be done according to the application requirement. For Example: see Listing 7.

If the keys are not declared with null character appended at the end, it will result in a bug. It affects the generation of XML format. So these types of developmental errors should be avoided.

The security impact is high as it provides the working flow of various structures that are used in web applications. It also provides information on the vulnerable function and the type of arguments passed. The attacker can use a trial and error mechanism to check the robustness of the function. The vulnerable function can be checked against buffer overflows. It depends on the type of arguments and initialized variables. Due to this factor, the security of web applications is impeded. An attacker can simultaneously design a function on the same pattern as the vulnerable function to test the insecure vectors. A simple vulnerable function not only lowers the effectiveness of an application but also the security parameters. Furthermore, the function can be used to leverage information extensively.

### XML Parsing Errors

Parsing errors are quite common in RSS based web applications. Usually XML based documents work on the root element specification. The working of elements and the benchmarks are provided by the W3C. The validation and specification is checked against the standards directly from the W3C website. It defines the usage of elements in a hierarchical way from the root to

the nodes. This type of error is basically a starting tag problem. The developers sometimes use special characters in the tags that cause parsing problems. The basic issue is that feeds are generated in a specific pattern already designed. If a certain code is prone to an error in the beginning, the feeds are not generated in the right manner thereby affecting the functionality of the RSS application extensively. See Listing 8.

Another reason for errors can be the mismatched version of the tags. Sometimes a developer forgets to use the end tags which are imperative for the completion of an element. For the start tag specification, the benchmark is stated here: <http://www.w3.org/TR/REC-xml/#sec-starttags>

These types of errors can be reduced to some extent by using XML Marker Editors.

So XML marking should be done in a correct manner to avoid errors. The depth of XML hierarchy can be extracted from the information resulted out. This not only shows the type of objects used but also the interdependencies and working usage of each object. It has been noticed in many web application configurations that the limit value for the XML hierarchy is low. For Example: The XML parsing vulnerabilities stated as:

```
http://publib.boulder.ibm.com/
infocenter/wasinfo/v4r0/index.jsp?topic=/
com.ibm.support.was40.doc/html/Security/
swg24003729.html
```

The impact of this action results in denial of service internally because elements will not be allocated after limit. In order to avoid this, the limit should be not be defined. The document has to be made dynamic for any number of object allocations. A better approach is

to use XML Markers for checking XML tag attributes. See Figure 3.

## Database Insecure Coding Checks: SQL Injection Base

Most of the applications designed for dynamic working in PHP have a database present in the backend. The database application has structured queries that are called by the user through the interface provided. The web three tier architecture works on this platform. A user simply provides input if required, or else the backend operations are executed automatically to get the work done.

The selection of variables and setting of parameters play a crucial role in the robust functionality of RSS based web applications. Let's have a look at the error code encountered in rss.php of some websites. See Listing 9.

### Listing 11. Error Due to a Path Problem in rss.php

```
Warning: main(../../../../includes/head_help.php): failed to open stream: No such file or directory in /vol/2/htdocs/blastmob/mobile/help/gen/rss.php on line 11

Warning: main(): Failed opening '../../../../includes/head_help.php' for inclusion (include_path='.:usr/local/netomat/php/lib/php') in /vol/2/htdocs/blastmob/mobile/help/gen/rss.php on line 11

Warning: main(../../../../includes/vars..php): failed to open stream: No such file or directory in /vol/htdocs/blastmob/mobile/help/gen/rss.php on line 11

Warning: main(): Failed opening '../../../../includes/vars..php' for inclusion (include_path='.:usr/local/netomat/php/lib/php') in /vol/2/htdocs/blastmob/mobile/help/gen/rss.php on line 11

Warning: main(../../../../includes/bodyStart_dynamic.php): failed to open stream: No such file or directory in /vol/2/htdocs/blastmob/mobile/help/gen/rss.php on line 16

Warning: main(): Failed opening '../../../../includes/bodyStart_dynamic.php' for inclusion (include_path='.:usr/local/netomat/php/lib/php') in /vol/2/htdocs/blastmob/mobile/help/gen/rss.php on line 16

Warning: main(../../../../includes/topNav.php): failed to open stream: No such file or directory in /vol/2/htdocs/blastmob/mobile/help/gen/rss.php on line 17

Warning: main(): Failed opening '../../../../includes/topNav.php' for inclusion (include_path='.:usr/local/netomat/php/lib/php') in /vol/2/htdocs/blastmob/mobile/help/gen/rss.php on line 17

Warning: main(../../../../includes/header.php): failed to open stream: No such file or directory in /vol/2/htdocs/blastmob/mobile/help/gen/rss.php on line 18

Warning: main(): Failed opening '../../../../includes/header.php' for inclusion (include_path='.:usr/local/netomat/php/lib/php') in /vol/2/htdocs/blastmob/mobile/help/gen/rss.php on line 18

RSS
```

### Listing 12. Error Prone Output

```
Warning: main(/home.10/paxatago/www/inc/prepend.php) [function.main]: failed to open stream: No such file or directory in /home.10/www/rss.php on line 34

Fatal error: main() [function.require]: Failed opening required '/home.10/www/inc/prepend.php' (include_path='.:usr/local/lib/php') in /home.10/www/rss.php on line 34
```



The above example is a direct outcome of bad coding. The error is not due to the result of the functions or arguments passed. The query that is structured to configure the database is not initialized generically. The declaration is not done effectively. The `iUserID` parameter is getting `NULL` causing the web application to show an exception during run time. The trace of a generated error is presented as such. The parameter is defined as `iUserID`. so it must be unique and cannot be null if no exception code is set. The developer has not designed an exception code to handle errors when `ID` is `NULL`. So it becomes a problem of coding again. These types of errors make the database unrecoverable if a proper backup is not made. Cross references defined for database optimization are the major cause of this happening. A single unstructured error can dismantle the proper functioning of the database. The problem can be reduced by setting debug code or exception handlers for tracing errors in RSS based web applications. Another point which should be taken into account while tracing errors is that the error information should not be displayed to the end user. There should be an error logging mechanism. Web administrators and developers should emphasize the importance of this factor. See Listing 10.

So this type of practice can avert the coding problems in web based database applications to some extent.

It affects the security element a lot. The leaking of database information acts as a basis for severe SQL injections. The attacker can easily extract the pattern of queries that are executed in the database. It not only provides the query information but also the objects and arguments to be supplied. The attacker can easily build new queries on the same pattern with different arguments to test the robustness of the application. The database can be updated very easily from a security point of view. The blind SQL injections can be run based on SQL information. The content manipulation inference attack can be performed very easily by rendering the response code of the web server to a constant value. One can test the application by the process of parameter splitting and balancing. Once the internal interface of the database

with a running application is breached, it becomes easy to perform data mining for extracting more information. It acts as an evolving chain process (chain reaction?) and web applications can be exploited through database injections. This in turn supports the theme of a vulnerability finding. A simple error in RSS database results in a full compromise.

## Path Configuration Checks – Missing Files or Directories

The configuration of files is a critical aspect as it influences the overall functionality of the software. After looking at a number of error prone `rss.php` pages, it has been found that the path of important files is not configured properly. The applications explicitly written for RSS checks require a number of extra files that support the run time execution. So those files have to be included in the code with definitive parameters. The PHP compiler requires proper paths where the libraries are located. This problem occurs mainly during installation when developers change the base directory and do not alter the core configuration files with respect to it. The problem is an intrinsic one but the errors are generated at the application level thereby preventing the interface from working. The cause of this type of problem is mismatch in the usage of extensions. The developers use certain extra functions which are not present in the normal extensions by default. It means the extensions have to be included externally and the path has to be specified in the configuration file. If this is not done effectively the application shows undesired behavior

Let's have a look at the error due to a path problem in `rss.php`. See Listing 11.

Another Error Prone Output: see Listing 12.

The above presented errors clearly demonstrate the point discussed above. The error pages are comprised in the path errors that originate from the misconfiguration of parameters. The configuration is an important part of development and should be done in a proper manner.

These types of errors reveal information which can be used to launch directory traversal attacks and

show web server objects in light of their security relevance. Generically, the directory structure on the web server can be understood. The path environment object shows internal information in the web directory and the way objects are organized. It is inevitable that error generation leads to information leakage. The information can be exploited by a malicious user to dig deeper into a web application and to learn the type of system objects participating in that interaction. The path disclosure is one of the main problems because it shows the hierarchy of the directory in which files reside. It is further used for directory traversal attacks if any misconfiguration is present in the application software.

## Conclusion

This paper elucidates a number of developmental problems that affect the robustness of Really Simple Syndication, i.e., RSS based web applications. Most of these applications are written in PHP with simulation of XML. The major point discussed is insecure vectors of coding and the problematic concerns they create. The code has to be verified both offline and online to prevent lapses. Continuous, persistent errors will result in vulnerabilities. Secure coding is the key solution to these problems. The emphasis is to implement security in hard core applications

## Note:

The underlined code gives you an idea of the conversion mechanism used to transport and export RSS feeds. The ASP and PHP code is presented. It is very crucial from the security point of view to actualize the source code for better understanding and testing of applications.

---

### Aditya K. Sood

Aditya K. Sood is an independent security researcher and founder of SecNiche Security. His online handle is 0kn0ck. He holds a BE and a MS in Cyber Law and Information Security. He is an active speaker at conferences like XCON, OWASP, and CERT-IN. His research interests include penetration testing, reverse engineering and web application security. Aditya's research has been featured in the USENIX login. Aditya's research projects include CERA Cutting Edge Research Analysis on Web Application Security (<http://www.ceras.ecniche.org>) and Mlabs featuring incore research (<http://mlabs.secniche.org>). The penetration testing issues are structured under TrioSec project. (<http://www.triosecc.secniche.org>). For regular updates on his work visit:  
<http://www.secniche.org>  
<http://zeroknock.blogspot.com>