MWR InfoSecurity Security Advisory

Plogger
SQL Injection Vulnerability

17th December 2007

MWR InfoSecurity

MWR INFOSECURITY

Contents

Plogger SQL Injection Vulnerability

Package Name:	Plogger
Date:	2007-10-15
Affected Versions:	Version 1.0 Beta 3.0

CVE Reference	Not Yet Assigned.
Author	D. Martin Gomez
Date	23rd October 2007
Severity	High.
Local/Remote	Remote.
Vulnerability Class	SQL Injection.
Vendor URL	http://www.plogger.org/
Version	Version 1.0 Beta 3.0
Vendor Response	The vendor has been contacted on 2007-10-29. Issue was addressed on changeset 489 (http://dev.plogger.org/changeset/489).
Exploit Details Included	Yes, although no exploit code is included.
Affected OS	The vulnerability has been confirmed on the Linux OS; however, all platforms are expected to be affected.

Overview:

An SQL injection vulnerability was identified in Plogger, a popular open source PHP photo gallery.

Impact:

The vulnerability would enable an attacker to inject arbitrary SQL statements. SQL injection inference techniques were used to develop a proof of concept exploit that could be used to access any field from the Plogger database (and potentially any field of any database accessible by the database user Plogger is configured to use).

Cause:

The Plogger RSS generation script (plog-rss.php) does not properly sanitise its input arguments.

Interim Workaround:

A patch is provided with this document.

Solution:

The affected software should be upgraded to the secure version available on the vendor's web site.

1 Detailed Vulnerability Description

1.1 Introduction

Plogger is an open source PHP photo gallery with over two years of development and more than 50,000 downloads. The Plogger web site (http://www.plogger.org), describes the application as a fully featured photo sharing package with an attractive and easy to use administrative interface.

1.2 Technical Background

SQL Injection is the process of injecting SQL commands into strings processed by an application. This is possible when there is insufficient validation of user input before it is executed in dynamic SQL queries.

As stated in "Data-mining with SQL Injection and Inference" (David Litchfield, September 2005), three classes of SQL injection attacks exist: in-band, out-of-band and the inference attack. In-band attacks extract data over the same channel between the client and the web server, for example, results are embedded in a web page via a union select. Out-of-band attacks employ a different communications channel to drill for data by using database mail or HTTP functions for example. Inference attacks stand alone in the fact that no actual data is transferred - rather, a difference in the way an application behaves can allow an attacker to infer the value of the data.

Typically, an attacker would initially look to exploit an SQL injection vulnerability by getting the results in-band. However, as described in the next section, this is not always possible and out-of-band or inference techniques are then required.

1.3 Vulnerability Details

It was found that insufficient validation was applied to the input parameters of the script that generates Plogger's RSS feeds. As a result, SQL code could be injected into Plogger database queries.

The vulnerability results from the following PHP code:-

```
$id = isset($_GET["id"]) ? $_GET["id"] : "";
```

As can be observed, the value of the **id** parameter is fetched and no input validation is performed. This value is then passed to another function that includes it as part of a database query.

Further investigation revealed that the injected code would be executed in two separate queries:-

Query 1

```
SELECT COUNT(DISTINCT p.`id`) AS cnt
      FROM `plogger_pictures` `p`
      LEFT JOIN `plogger_comments` `c` ON `p`.`id`=`c`.`parent_id`
      WHERE p.`parent_collection` = <injected code>
```

Query 2

```
SELECT p.*,
      UNIX_TIMESTAMP(`date_submitted`) AS `unix_date_submitted`,
      UNIX_TIMESTAMP(`EXIF_date_taken`) AS `unix_exif_date_taken`,
      COUNT(`comment`) AS `num_comments`
      FROM `plogger_pictures` `p`
      LEFT JOIN `plogger_comments` `c` ON `p`.`id`=`c`.`parent_id`
      WHERE p.`parent_collection` = <injected code>
      GROUP BY p.`id` ORDER BY `id`  DESC,p.`id` DESC  LIMIT 0,15
```

These two statements are completely different and although it is possible that an SQL statement could be found that would fit both and would deliver the desired output in-band, other techniques were found to be more appropriate in this case.

1.4 Exploit Information

Depending on the injected code, the server will return an error from either the first or the second query, or the queries will execute cleanly with no errors at all. This is an ideal scenario for SQL Injection inference attacks.

Source code inspection revealed that the first query is used to gather the number of images available in the database. This can be used to craft an exploit by inspecting interesting fields in the database and altering the script output depending on these values. For example, the results of the first query could be arranged to contain either a zero or the real number of available images. This, of course, would affect the output of the script. In one case, the RSS feed would contain no images, in the other, it would contain the correct number.

The existence of this vulnerability has been confirmed by MWR InfoSecurity and working exploit code that would allow database information gathering exists although this will not be released into the public domain at the present time. The decision to release such code in the future will be taken based on MWR InfoSecurity's obligations to protect its customers and Critical National Infrastructure (CNI) whilst also enabling the security community to accurately assess the vulnerability of systems running the software.

1.5 Dependencies

To exploit this vulnerability it is necessary to make a GET request to the web server. Therefore, network filtering can be put in place to protect a server in sensitive environments. However, it is accepted that as web servers are designed to be publicly accessible this mitigation will not be possible in the majority of circumstances.

In order to successfully exploit the attack vector described at least one picture needs to exist in the Plogger gallery.

2 Recommendations

It is recommended that all installations of the software be upgraded to the secure version available for download from the vendor's web site. However as an interim workaround the source code of **plog-rss.php** (line 103) could be patched like this:

```
$id = isset($_GET["id"]) ? intval($_GET["id"]) : "";
```

To reduce the level of risk to which users of the software are exposed it is further advised that the application be run under a database user account with the lowest level of privilege possible.

It is also recommended that, where possible, the Plogger photo gallery should be subject to network level filtering such that only trusted IP addresses can communicate with the service. It should be noted that this is a generic recommendation and is not specific to this technology.

3 References

- "Data-mining with SQL Injection and Inference" (David Litchfield, September 2005).

http://www.ngssoftware.com/research/papers/sqlinference.pdf

MWR InfoSecurity
St. Clement House
1-3 Alencon Link
Basingstoke, RG21 7SB
Tel: +44 (0)1256 300920
Fax: +44 (0)1256 844083
mwrinfosecurity.com