Penetration Test:
A Case Study on Remote Command Execution Security Hole

Dr. S.Mohammad[a], Soulmaz Pourdavar[b]

[a,b]Department of Industry, IT Group, K.n.Toosi University of Technology,Tehran, Iran
[a]smohammadi40@yahoo.com
[b]pourdavar@gmail.com

Abstract

This paper offers a fresh perspective on the aspect of application security, highlighting a sample attack that is not currently being protected against. Here is a case study which discussed identifying poor coding practices that render Web applications vulnerable to attacks such as remote command execution. Given the increased focus on the need for application security, it is now to be hoped that the issue will receive greater attention in new software releases. In this research a case study is discussed on the basilic software which has a great usability in the publication and educational web sites in Europe. Although this is a useful software, the research identified some security holes on the application and offers a proof on vulnerability of the software and a solution for this problem is explained.

Keywords: Penetration Test, Security hole, Remote Command Execution, Vulnerability.

1. Introduction

Most organizations recognize the importance of cyber security and are implementing various forms of protection. However, many are failing to find and fix known security problems in the software packages they use as the building blocks of their networks and systems, a vulnerability that a hacker can exploit to bypass all other efforts to secure the enterprise [1]**.** In this paper a case study is discussed which is about a remote command execution hole that found on a web application. Part two is about the description and the importance of the selected application for study about, part three and four are some discussion about vulnerability and penetration test, Part five is about Remote Command Execution Security Hole, part six addressed the problem on case study's software and there are some discussion on the proof of concept, part seven represent a solution with a simulation to proof of concept, part eight offers some tips to preventing such problems and part eight is the conclusion of the research.

1. The Importance of selected software

1.1. Basilic 1.5.14 description

This software is installed on so many web sites and provides a bibliography server for research laboratories. It automates and facilitates the diffusion of research publications over the Internet, automatically generating Web pages from a publication database. Each publication has an associated Web page, which provides downloads and additional documents (abstract, images, BibTeX). Index pages are also created, including a search engine with several options for results display. New publications can be added to the database in an instant [2].

Here is a part of the application's user guide:
BibTex-like publication records (title, authors, year, journal, ...) are stored in a database, and web pages are automatically generated from the database using php scripts. Publications can be added, edited or deleted using an intranet-based back-office interface. All local users have access to all publications - publications are not managed by their "owner" or by a dedicated administrator. This policy can of course be changed, but this freedom ensures an up-to-date server and presents no problems in practice.

Each publication has its own associated directory, located in the public (web browseable) section of the server. The directories are created automatically when publications are added and include an index.php file which is the publication associated web page. These directories are organized by year, and have names generated from the initials of the authors' surnames, such as ABC0. The publication's authors have a write access to these directories and can place there the documents associated with the publication. These will automatically be parsed and referenced by the publication's associated web page. This is a key feature of the Basilic server: simply

move files into the right directory to make them available on the web[2].

2. Vulnerability (Security Flaw)

In computer security, the term vulnerability is a weakness which allows an attacker to reduce a system's Information Assurance. Vulnerability is the intersection of three elements: a system susceptibility or flaw, attacker access to the flaw, and attacker capability to exploit the flaw [3]. Using the failure of the system to violate the site security policy is called exploiting the vulnerability. To be vulnerable, an attacker must have at least one applicable tool or technique that can connect to a system weakness. In this frame, vulnerability is also known as the attack surface.

A vulnerability with one or more known instances of working and fully-implemented attacks is classified as an exploit. The window of vulnerability is the time from when the security hole was introduced or manifested in deployed software, to when access was removed, a security fix was available/ deployed, or the attacker was disabled.

Constructs in programming languages that are difficult to use properly can be a large source of vulnerabilities [4].

2.1 Security bug

A security bug is a software bug that benefits someone other than intended beneficiaries in the intended ways[5].

3. Penetration test

The goal of security penetration services is to help organizations secure their systems. Penetration Study is a test for evaluating the strengths of all security controls on the computer system. It intends to find all possible security holes and provides suggestions for fixing them.

Penetration Testing is an authorized attempt to violate specific constraints stated in the form of a security or integrity policy. Example goals of penetration studies are gaining of read or write access to specific objects, files, or accounts, gaining of specific privileges and disruption or denial of the availability of objects.

Penetration Testing is a testing technique for discovering, understanding, and documenting all the security holes that can be found in a system. It is not a proof technique. It can never prove the absence of security flaws. It can only prove their presence.

A more thorough penetration study is to find the proper interpretation of vulnerabilities found, draw conclusion on the care taken in the design and implementation. Testers should know that making a simple list of vulnerabilities, although helpful in closing those specific holes, contributes far less to the security of a system. In practice, constrains (resource, money, time) affect the penetration study.

There are two kinds of penetration Tests: Announced and Unannounced. Although Announced testing is Efficient and Team oriented, The Holes may be fixed as discovered and it causes false sense of security. Unannounced testing has a Greater range of testing but Response may block further penetration, requires strict escalation process and Impact operations. Attackers for collect information about the target act in several ways: External attacker with no knowledge of the system, External attacker with access to the system, internal attacker with access to the system [9].

4. Remote command execution security hole

There are 3 security holes that allow attacker to upload shell to the web site: Remote command execution, Remote File Inclusion and Framework [6]. In this paper the focus is on the first security hole.

5. The problem – Case Study on Basilic Software

During the research for the paper a number of websites were identified as being vulnerable to Remote Command Execution. Specific details for one of these holes are given below together with an example of how an attack could be performed. A patch exists that resolves this issue, details of this can be found in the latter of the text.

5.1 Basilic files

Basilic software is composed of some file and folders. As shown below the focus is on the php files and their directories.

- Folders
  - Config
    - checkConfig.php
    - diff.php
    - include.php
  - CSS
  - Images
  - Import

- Intranet
- Public
- Sources
  - Intranet
    - Authors
      - author.php
      - authorAction.php
      - menuAuthor.php
    - Publications
      - menuPubli.php
      - publi.php
      - pubAction.php
      - updatePublis.php
    - utils.php
    - updatePubdocs.php

The file diff.php in config folder has the security hole called Remote Command Execution.

5.2. Proof of concept

An attacker may perform an exploit, here are the examples of sending DOS commands to the victim website which is using basilic software version 1.5.14 via a DOS shell and the website's replies. The shell is written in Perl programming language:

```
C:\Perl\bin>perl basilic.pl
www.xxx.xxx  80  /publications/
<SHELL>dir
```

And this information appears from the victim site which is the current directory files:

checkConfig.php,include.php,install.html, tables.txt, diff.php, index html

```
<SHELL>ls -la

total 424
drwxrwxr-x  3 micc     www-data   4096 Dec 27 10:27 .
drwxrwxr-x 28 micc     www-data   4096 Dec  7 10:45 ..
-rw-rw-r--  1 micc     www-data  20527 Mar 31  2008 checkConfig.php
-rw-rw-r--  1 micc     www-data   1130 Jul 27 14:47 diff.php
-rw-rw-r--  1 micc     www-data     32 Mar 31  2008 include.php
-rw-r--r--  1 www-data www-data     18 Nov  4 13:19 index.html
-rw-rw-r--  1 micc     www-data  12454 Mar 31  2008 install.html
-rw-rw-r--  1 micc     www-data   3658 Mar 31  2008 tables.txt


C:\Perl\bin>basilic.pl   www.xxx.xxx    80 /publications/
<SHELL>pwd

/home/micc/public_html/publications/Config
```

5.3. Where is the problem?

The problem caused from part of a php file, named diff.php that finds the difference between an old file and its last updated version. Here is a part of this file:

```
<?php
If  (empty($_GET["file"]))
{
 Die("No  <code>file</code>  provided");
}
If  (empty($_GET["old"]))
{
 Die("No  <code>old</code>  directory  provided");
}
If  (empty($_GET["new"]))
{
 Die ("No <code>new</code> directory  provided");
}
Echo "The old file is <code>$_GET[old]
/$_GET[file] </code> <br/>\n";

Echo "The new file is <code>$_GET[new]
/$_GET[file] </code> <br/>\n";

Echo "Here is the diff between the two files
:\n<pre>\n";

System ("diff ../$_GET[old]/$_GET[file]
$  GET[new]/$  GET[file] | sed s%\"<\"%\"\&it;\"%g
| sed s%\">\"%\"\&gt;\"%g");

Echo  "\n</pre>\n<p><a  href=\
"../checkConfig.php\">back</a></p>\n";
?>
```

This part on the line number 39 has caused the hole:

| sed s%\"<\"%\"\<\"%g | sed s%\">\"%\"

6. The solution

For patching the file this part should be such:

```
$old=escapeshellarg($_GET["old"]);

$file=escapeshellarg($_GET["file"]);

$new=escapeshellarg($_GET["new"];

System("diff  ../$old/$file  $new/$file  |  sed s% |
\"<\"%\"\&it;\"%g  sed s%\">\"%\"\&gt;\"%g");
```

The escapeshellarg() method prevent from accepting some characters such as &, ; and $ that could be sent from attackers, and the placement of the pipe character (|) has been changed.

6.1. Proof

For simulating the attack to the web site before and after the patching of application, the wamp software is chosen, which provides a web server without installing the apache, php or mysql server on the pc. Then basilic application's folder is pasted on a folder of wamp server named www.
Here is the examination in the
http://127.0.0.1/basilic/config/diff.php?file=1&old=1&new=%26dir%26

The reply from the website is:

The old file is 1/1
The new file is & dir&/1
Here is the diff between the two files:

Volume in drive C has no label .
Volume serial Number is xxxx-xxxx

Directory of C:\wamp\www\basilic\Config

```
                ..
05/02/2007 11:19 PM 20.52 checkConfig.php
05/02/2007 11:19  PM   1.130 diff.php
05/02/2007 11:19  PM   32      include.php
05/02/2007 11:19  PM   12.45 install html
05/02/2007 11:19  PM   3.658 tables.txt

      5 File(s)      37.801 bytes
      2 Dir(s)       7.241.859.072 bytes free
```

After patching the application the reply of this request is shown below:

The old file is &dir&/1
The new file is 1/1
Here is the diff between the two files:...

And the website did not show any information unless the difference of the two files.

7. Preventing code injection

To prevent code injection problems, utilize secure input and output handling and some common tips such as:

- Install only required softwares, open only required ports.
- Input validation
- Escaping dangerous characters. For instance, in PHP, using the htmlentities () function to protect general inputs into the web application, and mysql_real_escape _string() to protect inputs which will be included in an SQL request, to protect against SQL Injection.
- Input encoding
- Output encoding
- Other coding practices which are not prone to code injection vulnerabilities, such as "parameterized SQL queries" (also known as "prepared statements" and sometimes "bind variables").
- Modular shell disassociation from kernel
- Update latest patches
- Change default settings/options
- Setup password and protect your password file.
- Install anti-virus software and keep it updated.
- Maintain a good backup.
- Have a good emergency plan.
- Periodic monitor of the system.
- Have a good administrator.

8. Conclusion

In this study a security hole in a web application software named basilic 1.5.14 was found. The study proved that this problem exists on this version of software, which allow the remote command execution attack to the web server. Then we show that the problem is raised from the line 39 in a php file (diff.php) on the "config" folder. The paper offered a solution on this problem and proved that the proposed way of coding could prevent from such attacks.
It should be noted that the exploit technique discussed is not new; however, the abuse of these techniques and how this applies to web applications has not been widely discussed in the public domain. Website administrators must consider the vulnerabilities of the remote command execution holes and various troubles which could occur via hackers, and vendors should pay more attention on the applications' security. This wouldn't occur unless the programmers accustomed to use secure techniques of design and coding as an essential part of their job.

Further researches may follow the correction of weaknesses uncovered by the penetration exercise, Automate and customize the penetration test process, Use of intrusion detection systems, Use of honeypots and honeynets.

7. References

[1] Martin,R.A., “Managing vulnerabilities in networked systems”, IEEE.org, Nov 2001.

[2] http://artis.imag.fr/Software/Basilic.

[3] Rafael Dominguez Vega, “Behind Enemy Lines”, mwrinfosecurity, St. Clement House- 1-3 Alencon Link-Basingstoke-RG21 7SB, Tel: +44 (0)1256 300920.

[4] U.S. Air Force Software Protection Initiative, “The Three Tenents of Cyber Security".

http://www.spi.dod mil/tenets.htm. Retrieved 2009-12-15.

[5] http://en.wikipedia.org/wiki/Security_bugs.

[6] Dr.Hedaya Alasooly, "Hacking Tools", Ministry of Telecom and Inform Tech, Palestine, hasooly@gov.ps.J. Wang, "Fundamentals of erbium-doped fiber amplifiers arrays" (Periodical style—Submitted for publication), IEEE J. Quantum Electron., submitted for publication.

[7] http://en.wikipedia.org/wiki/Code_injection.

[8] Klevinsky, et. al. "Hack I.T.-Security Through Penetration Testing". ISBN 0-201-71956-8.

[9] Paul Fong & Cai Yu, CS691, "Penetration Testing & Countermeasures", 5 May 2003.

[10] Information Security & Network Research Group, School of Computing, Communications & Electronics, University of Plymouth, lymouth, UK. Elsevier Ltd. 2007.