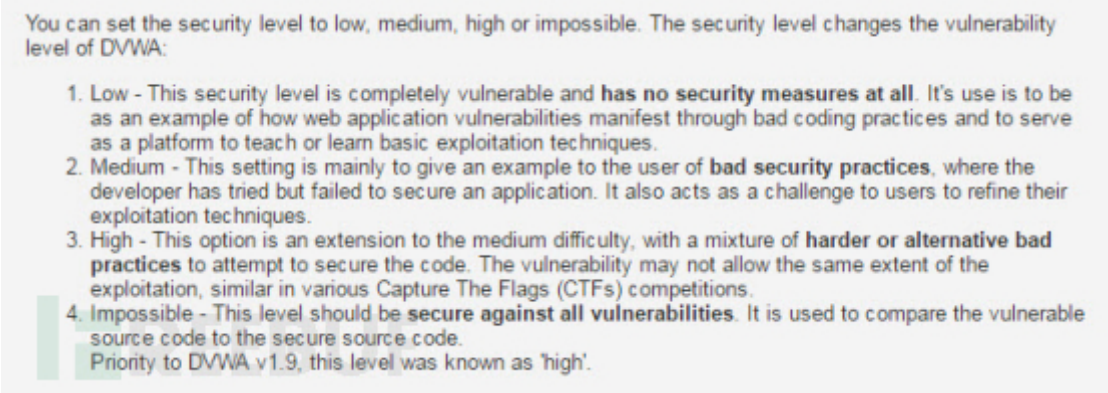
目前，最新的DVWA已经更新到1.9版本（http://www.dvwa.co.uk/），而网上的教程大多停留在旧版本，且没有针对DVWA high级别的教程，因此萌发了一个撰写新手教程的想法，错误的地方还请大家指正。

DVWA简介

DVWA（Damn Vulnerable Web Application）是一个用来进行安全脆弱性鉴定的PHP/MySQL Web应用，旨在为安全专业人员测试自己的专业技能和工具提供合法的环境，帮助web开发者更好的理解web应用安全防范的过程。

DVWA共有十个模块，分别是Brute Force（暴力（破解））、Command Injection（命令行注入）、CSRF（跨站请求伪造）、File Inclusion（文件包含）、File Upload（文件上传）、Insecure CAPTCHA（不安全的验证码）、SQL Injection（SQL注入）、SQL Injection（Blind）（SQL盲注）、XSS（Reflected）（反射型跨站脚本）、XSS（Stored）（存储型跨站脚本）。

需要注意的是，DVWA 1.9的代码分为四种安全级别：Low，Medium，High，Impossible。初学者可以通过比较四种级别的代码，接触到一些PHP代码审计的内容。


You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
   Priority to DVWA v1.9, this level was known as 'high'.

DVWA的搭建

Freebuf上的这篇文章《新手指南：手把手教你如何搭建自己的渗透测试环境》（http://www.freebuf.com/sectool/102661.html）已经写得非常好了，在这里就不赘述了。

之前介绍了Brute Force模块的内容（http://www.freebuf.com/articles/web/116437.html），本文介绍的是Command Injection模块，后续教程会在之后的文章中给出。

Command Injection

Command Injection，即命令注入，是指通过提交恶意构造的参数破坏命令语句结构，从而达到执行恶意命令的目的。PHP命令注入攻击漏洞是PHP应用程序中常见的脚本漏洞之一，国内著名的Web应用程序Discuz!、DedeCMS等都曾经存在过该类型漏洞。

下面对四种级别的代码进行分析。

Low

服务器端核心代码

```
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {

    // Get input

    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.

    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {

        // Windows


        $cmd = shell_exec( 'ping  ' . $target );

    }

    else {

        // *nix

        $cmd = shell_exec( 'ping  -c 4 ' . $target );

    }

    // Feedback for the end user

    echo "<pre>{$cmd}</pre>";

}

?>
```

相关函数介绍

stristr(string,search,before_search)

stristr
函数搜索字符串在另一字符串中的第一次出现，返回字符串的剩余部分（从匹配点），如果未找到所搜索的字符串，则返回
FALSE。参数string规定被搜索的字符串，参数search
规定要搜索的字符串（如果该参数是数字，则搜索匹配该数字对应的ASCII值的字符），可选参数before_true为布尔型，默认为“false”，如果设置为“true”，函数将返回search参数第一次出现之前的字符串部分。

php_uname(mode)

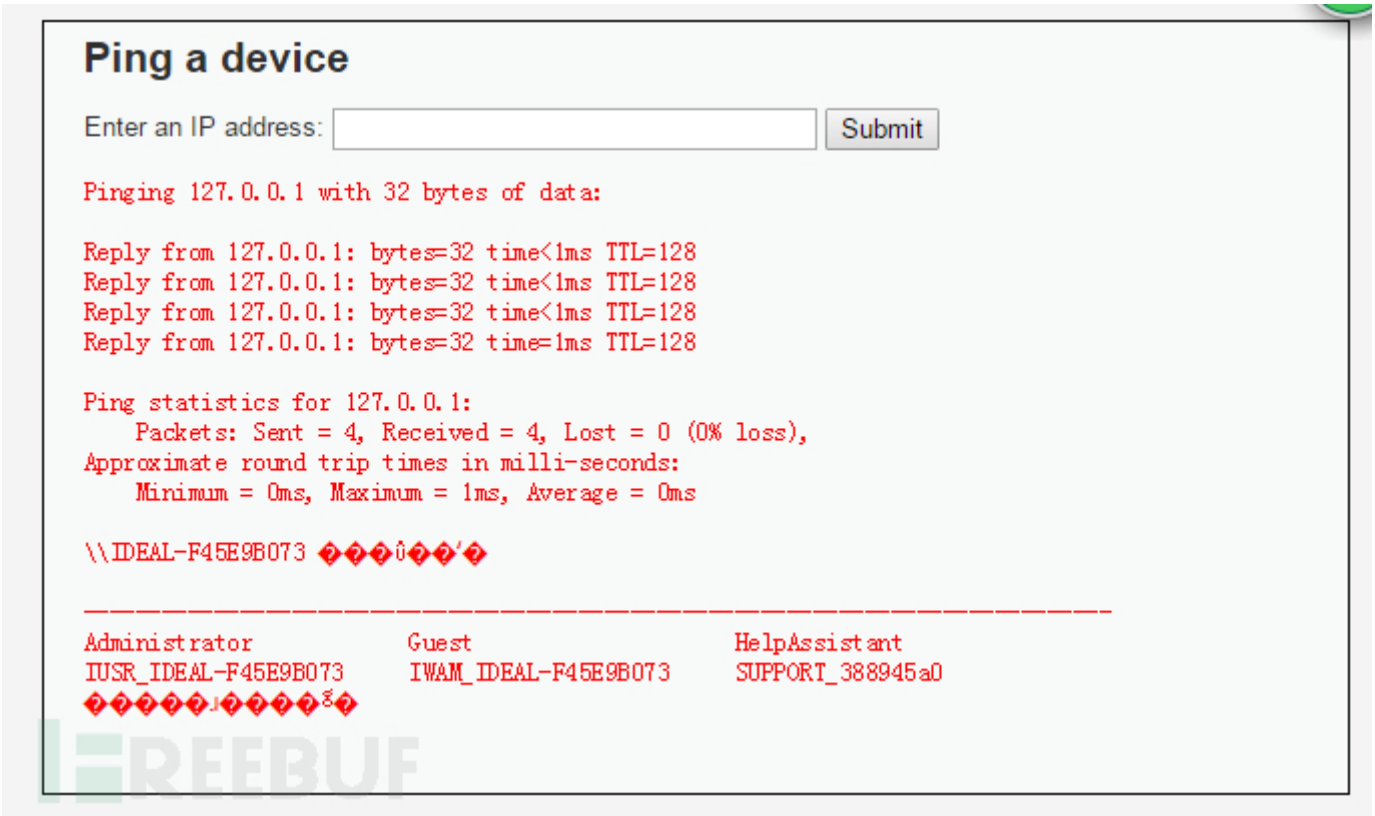
这个函数会返回运行php的操作系统的相关描述，参数mode可取值”a”（此为默认，包含序列”s n r v m”里的所有模式），”s”（返回操作系统名称），”n”（返回主机名），” r”（返回版本名称），”v”（返回版本信息），” m”（返回机器类型）。

可以看到，服务器通过判断操作系统执行不同ping命令，但是对ip
参数并未做任何的过滤，导致了严重的命令注入漏洞。

漏洞利用

window和linux系统都可以用&&来执行多条命令

127.0.0.1&&net user



Linux下输入127.0.0.1&&cat /etc/shadow甚至可以读取shadow文件，可见危害之大。

Medium

服务器端核心代码

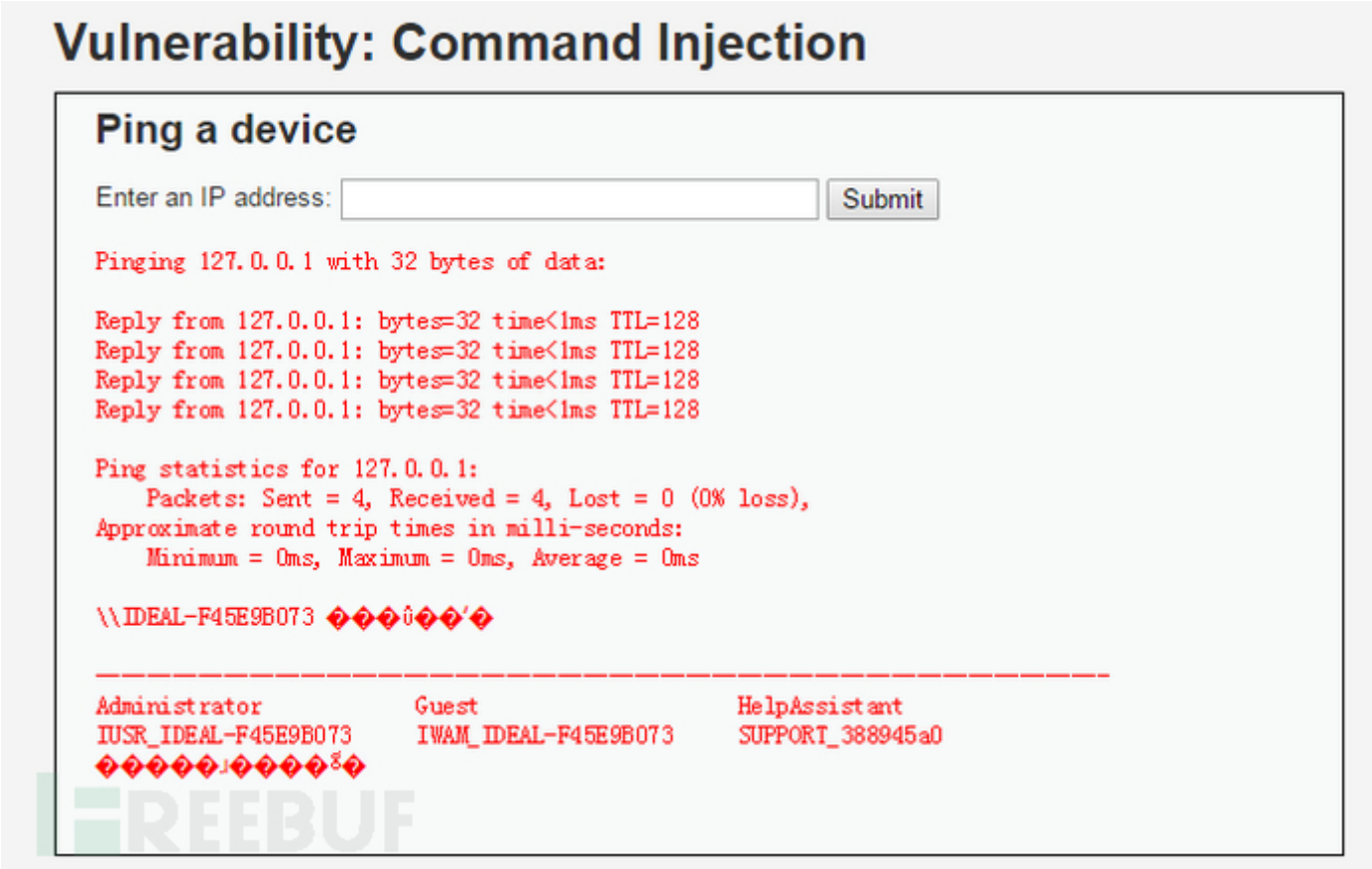
```
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {

    // Get input

    $target = $_REQUEST[ 'ip' ];

    // Set blacklist

    $substitutions = array(

        '&&' => '',

        ';'  => '',

    );

    // Remove any of the charactars in the array (blacklist).

    $target = str_replace( array_keys( $substitutions ), $substitutions, $target
);

    // Determine OS and execute the ping command.

    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {

        // Windows

        $cmd = shell_exec( 'ping  ' . $target );

    }

    else {

        // *nix

        $cmd = shell_exec( 'ping  -c 4 ' . $target );

    }

    // Feedback for the end user

    echo "<pre>{$cmd}</pre>";

}

?>
```

可以看到，相比Low级别的代码，服务器端对ip参数做了一定过滤，即把”&&“ 、”;“删除，本质上采用的是黑名单机制，因此依旧存在安全问题。

漏洞利用

1、127.0.0.1&net user

因为被过滤的只有”&&”与” ;” ，所以”&”不会受影响。



这里需要注意的是”&&”与” &”的区别：

Command 1&&Command 2

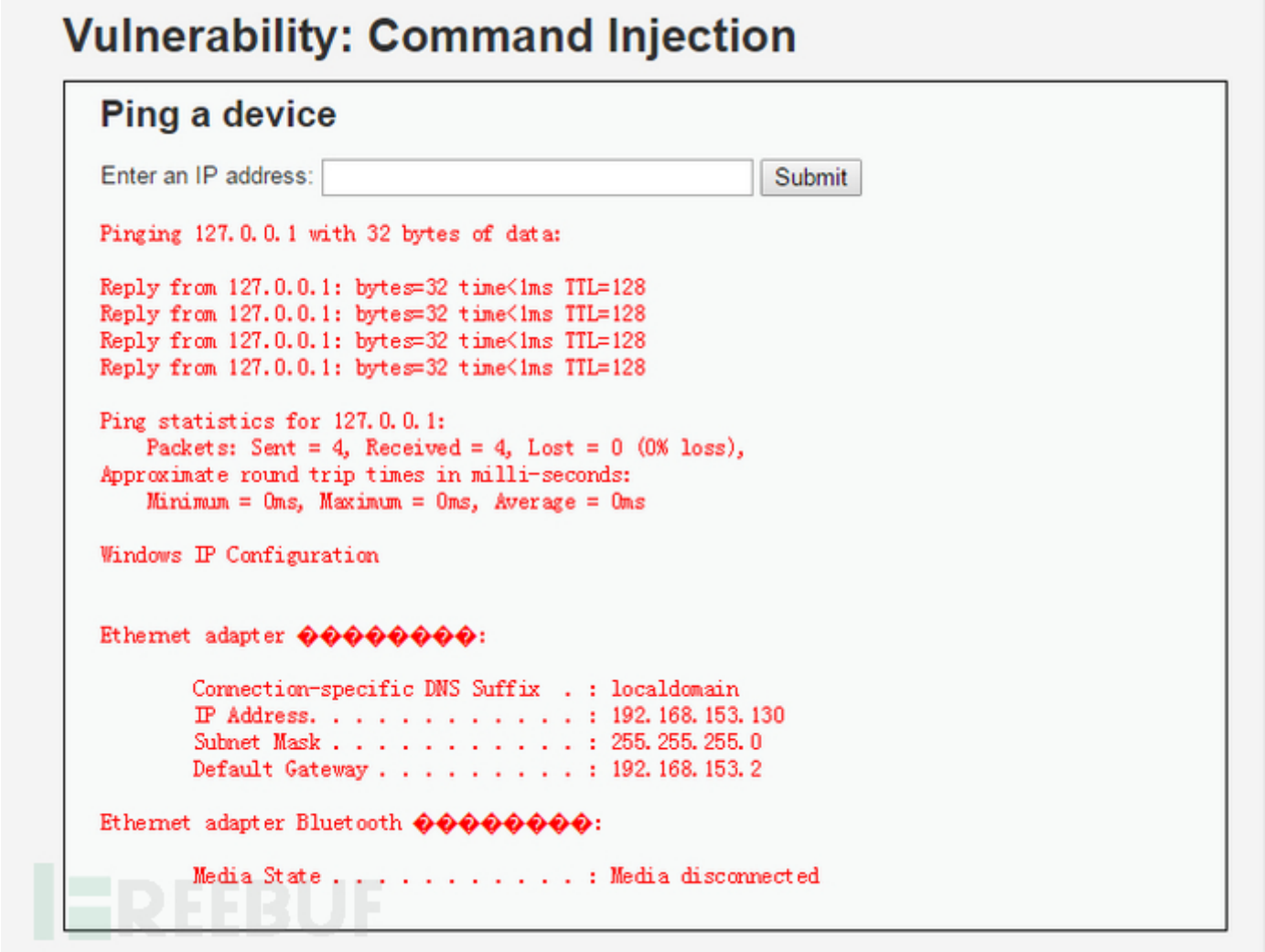
先执行Command 1，执行成功后执行Command 2，否则不执行Command 2

Command 1&Command 2

先执行Command 1，不管是否成功，都会执行Command 2

2、由于使用的是str_replace把”&&” 、”;”替换为空字符，因此可以采用以下方式绕过：

127.0.0.1&;&ipconfig



这是因为”127.0.0.1&;&ipconfig”中的” ;”会被替换为空字符，这样一来就变成了”127.0.0.1&&ipconfig” ，会成功执行。

High

服务器端核心代码

```php
<?php

if( isset( $_POST[ 'Submit' ]  ) ) {

    // Get input

    $target = trim($_REQUEST[ 'ip' ]);

    // Set blacklist

    $substitutions = array(

        '&'  => '',

        ';'  => '',

        '|  ' => '',

        '-'  => '',

        '$'  => '',

        '('  => '',

        ')'  => '',

        '`'  => '',

        '||' => '',

    );

    // Remove any of the charactars in the array (blacklist).

    $target = str_replace( array_keys( $substitutions ), $substitutions, $target
);

    // Determine OS and execute the ping command.

    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {

        // Windows

        $cmd = shell_exec( 'ping  ' . $target );

    }

    else {

        // *nix

        $cmd = shell_exec( 'ping  -c 4 ' . $target );

    }

    // Feedback for the end user
```

```
    echo "<pre>{$cmd}</pre>";
}
?>
```

相比Medium级别的代码，High
级别的代码进一步完善了黑名单，但由于黑名单机制的局限性，我们依然可以绕过。

漏洞利用

黑名单看似过滤了所有的非法字符，但仔细观察到是把“|”（注意这里|后有一个空格）替换为空字符，于是“|”成了“漏网之鱼”。

127.0.0.1|net user

Command 1 | Command 2

“|”是管道符，表示将Command 1的输出作为Command 2的输入，并且只打印Command 2执行的结果。

Impossible

服务器端核心代码

```
<?php
if( isset( $_POST[ 'Submit' ]  ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ],
'index.php' );
    // Get input
    $target = $_REQUEST[ 'ip' ];
    $target = stripslashes( $target );
```

```
    // Split the IP into 4 octects
    $octet = explode( ".", $target );
    // Check IF each octet is an integer
    if( ( is_numeric( $octet[0] ) ) && ( is_numeric( $octet[1] ) ) && ( is_numeric( $octet[2] ) ) && ( is_numeric( $octet[3] ) ) && ( sizeof( $octet ) == 4 ) ) {
        // If all 4 octets are int's put the IP back together.
        $target = $octet[0] . '.' . $octet[1] . '.' . $octet[2] . '.' . $octet[3];
        // Determine OS and execute the ping command.
        if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
            // Windows
            $cmd = shell_exec( 'ping  ' . $target );
        }
        else {
            // *nix
            $cmd = shell_exec( 'ping  -c 4 ' . $target );
        }
        // Feedback for the end user
        echo "<pre>{$cmd}</pre>";
    }
    else {
        // Ops. Let the user name theres a mistake
        echo '<pre>ERROR: You have entered an invalid IP.</pre>';
    }
}
// Generate Anti-CSRF token
generateSessionToken();
?>
```

相关函数介绍

stripslashes(string)

stripslashes函数会删除字符串string中的反斜杠，返回已剥离反斜杠的字符串。

explode(separator,string,limit)

把字符串打散为数组，返回字符串的数组。参数separator规定在哪里分割字符串，参数string
是要分割的字符串，可选参数limit规定所返回的数组元素的数目。

is_numeric(string)

检测string是否为数字或数字字符串，如果是返回TRUE，否则返回FALSE。

可以看到，Impossible级别的代码加入了Anti-CSRF token，同时对参数ip进行了严格的限制，只有诸如“数字.数字.数字.数字”的输入才会被接收执行，因此不存在命令注入漏洞。