/==W
----------------------------[PLAYHACK.net]-----------------------------
W==/

-[INFOS]---

Title: "PHP Undergroud Security"
Author: Omnipresent
E-Mail: omnipresent@email.it - omni@playhack.net
Website: http://omni.playhack.net - http://www.playhack.net
Date: 2007-04-12

번 역: bOBaNa (WOWHACKER.ORG)
메 일: hackprog@korea.ac.kr

- PHP 공부에 많은 도움이 되셨으면 합니다. 서투른 영어실력에 오역이 있을 수 있고요
  의역이 많습니다. 잘못된 곳이 있다면, 위 메일로 연락주시면 감사하겠습니다. ^^
  항상 즐거운 나날 되시길 바랍니다.   07/9/22

-[SUMMARY]---

---[0x00: Let's start..]

모두 안녕하세요, 먼저 모국어가 아니기 때문에 형편없는 제 영어에 죄송스럽다는 말을 전합니다.

이 튜토리얼에서는 중요한 PHP 실수에 대하여 설명드릴 것입니다. ; 어떻게 그것을 찾을 것인가, 끝으로 어떻게 익스플로잇을 하는지, 어떻게 패치를 하는지!

즐겨요!
---[/]

---[0x01: 전역변수를 주의하여 살펴봐요.]

PHP에서 변수를 선언하지 않아도 되며, 그건 프로그래머에게 반가운 일입니다.
사용자가 변수를 사용해야할 때 변수들은 "자동적으로" 만들어집니다. 좋은것으로 생각되죠. 그렇죠, 그렇지만, 여러분도 알다시피 PHP는 종종 사용자에 의해서 입력을 받고, 그것을 처리할 것입니다. 만약 우리가 변수를 사용하기 전에 변수를 선언하지 않아도 된다면, 웹어플리케이션의 보안에 대한 문제에 대해서 생각할 수 있습니다.
예를 들어서 다음 코드를 보죠. :

```
        [...]

        if ($is_admin == 1) {
                //그렇죠. 저는 관리자라고 불리는 admin입니다.
                [...]
        } else {
                //아니죠, 저는 admin이 아닙니다.
                [...]
        }
```

네, $is_admin은 이전에 사용된적이 없던 변수인걸 알 수 있으며, 그래서 만약,
우리가 이 변수를 PHP 스크립트(버그가 있는)로 넘길 수 있다면, 우리는 관리자로의
권한없는 접근을 할 수 있을 거에요. 그렇죠, 그런데 어떻게 Admin권한을 얻을까요?
간단해요:

```
http://remote_host/bugged.php?is_admin=1
```

---[Patching]---

이 버그를 어떻게 패치할까요? 간단하고 빠르게: IF 조건문이 나오기 전에 $is_admin을
선언합니다. 다음과 같이:

```
$is_admin = 0;

[...]

        if ($is_admin == 1) {
                //그렇죠, 저는 관리자라고 불리는 admin입니다.
                [...]
        } else {
                //No, 저는 admin이 아닙니다.
                [...]
        }
```

----------[/]

---[0x02: File Inclusion] // 파일 삽입

-----[Local File Inclusion]

PHP는 파워풀하고 간단한 좋은 언어에요. 그렇지만, 여러분이 작성한 코드에서의
보안이슈를 원하지 않다면, 반드시 주의해서 코드를 살펴보아야 합니다.

많은 경우에 경로의 이름(pathname)을 변수에 저장하여 사용하는 동적 삽입을 사용합니다.
예를들어, 다음 예제를 봅시다:

```
<?php

        include "/users/".$include_path.".php";
        [...]
?>
```

$include_path는 사용되기 전에 선언되지 않아서, 공격자는 이 변수에 변조된 데이터를
집어넣고 /etc/passwd와 같은 파일을 포함합니다. 예를 들어, 사용자는 "쉽게" URL에 있는
include_path의 변경되는 값을 통해 다른 파일을 볼 수 있습니다.

```
http://remote_host/bugged.php?include_path=../../../../etc/passwd%00
```

이 삽입의 결과는 다음과 같을 것입니다:

```
<?php

          include "/users/../../../../etc/passwd%00.php"
          [...]
?>
```

그렇게 악의있는 사용자는 서버에 저장된 모든 암호를 볼 수 있습니다.

- [NOTE] -

%00은 무엇인가요? 간단해요, %00은 PHP 확장자를 "삭제"하는 NULL CHAR를 의미합니다.
만약 이 NULL byte를 빼먹는다면, 오직 .php 파일만을 표시할 수 있을 거에요.
because the extension included is PHP
(include "/users/".$include_path.".PHP")

-----[Remote File Inclusion] // 원격 파일 삽입

다음의 코드를 살펴봅시다:

```
<?php

          [...]

                    include($_GET['pag']);

          [...]

?>
```

위에서 볼 수 있듯이, $page는 사용되기 전에 검증되지 않기 때문에, 악의있는 사용자가
브라우져를 통해 악성스크립트를 삽입하거나 호출할 수 있을 것입니다.

첫번째 예제: (머신으로의 접근을 얻는다)

http://remote_host/inc.php?pag=[악의적 스크립트 - 서버의 쉘 위치]

두번째 예제: (파일을 본다)

http://remote_host/inc.php?pag=/etc/passwd

---[Patching]---
입력을 확인하세요. 입력을 확인하는 많은 방법 중 하나는 아래 보여준것처럼 접근 가능한
페이지의 리스트를 만드는 겁니다 :

```
[...]

          $pag = $_GET['pag'];

          $pages = array('index.php', 'alfa.php', 'beta.php', 'gamma.php');

                    if(in_array($pag, $pages))
                    {
                              include($pag);
                    {
                              else
```

```
                        {
                        die("Hacking Attempt!");
                        }

[...]
```

--[/]

---[0x03: XSS]

XSS에 대한 것을 원하시나요?

저의 파트너는 XSS에 대해 좋은 일을 했습니다. 다음 문서를 보세요 :

[+] "Cross-Site Scripting for Fun and Profit"
    http://www.playhack.net/view.php?type=1&id=18

[+] "Applying XSS to Phishing Attacks"
    http://www.playhack.net/view.php?type=1&id=20

--[/]

---[0x03: SQL Injection]

SQL 인젝션은 그 이름이 말해주듯이, 여러분은 버그가 있는 웹 어플리케이션의 쿼리 스트링에 SQL코드를 삽입할 수 있습니다.

------[0x04b Login Bypass] // 로그인 우회

멋진(:D) 예제를 통해 시작하기 전에 SQL에 대해 어느정도는 알아야 합니다. :

- (') 정점? 그게 뭔가요? SQL 문장에서는 (')은 연산자이며, 우리에게는(공격자의 관점에서는) (')은 어떤 취약점을 익스플로잇하기에 아주 중요한 연산자입니다.

(')은 스트링의 범위를 정합니다..

- (#) 코멘트? 맞아요 :D '#'(정점을 제외하고)과 함께 SQL 문장에서 주석을 만들수 있습니다. 아주 유용하고 중요한 것이니 기억해 두세요!

- (;) (;)은 데이터베이스에 새로운 쿼리를 만드는 것을 말해요. 쉽죠.

약간의 유용한 피연산자들을 본 다음에 (맞아요. 해킹을 위해서죠 :D) 첫번째 예제를 만들 수 있어요 : 로그인 우회 - 관리자 권한을 얻는다
좋아요.

```
<?php

        // login.php

        $nick = $_POST['nick'];
        $pass = $_POST['pass'];

        $link = mysql_connect('localhost', 'root', 'root') or
                 die('Error: '.mysql_error());

        mysql_select_db("sql_inj", $link);
```

```
        $query = mysql_query("SELECT * FROM sql_inj
                              WHERE nick ='".$nick."'
                              AND pass ='" .$pass. "'", $link);

        if (mysql_num_rows($query) == 0) {
                 echo "<script type=W"text/javascriptW">
                        window.location.href='index.html';</script>";
        exit;
}

        $logged = 1;

        [...]

        //EoF

        ?>
```

이 스크립트는 아주 쉽지만, SQL 인젝션에 대해서 배우기는 매우 유용합니다.
앞서 볼 수 있듯이 변수 $nick과 $pack는 SQL 쿼리에 사용되기 전에 적절하게
처리되어 있지 않습니다(보안적으로), 그렇기 때문에 SQL 코드가 삽입될 수 있습니다.

다음 쿼리를 보죠:

```
"SELECT * FROM sql_inj WHERE nick ='".$nick."' AND pass ='" .$pass. "'"
```

아래와 같이 변조된 두 변수를 통과시키면 어떤 일이 일어날까요:

```
$nick = 1' OR '1' = '1
$pass = 1' OR '1' = '1
```

새로운 쿼리는 다음과 같을 거에요:

```
"SELECT * FROM sql_inj WHERE nick ='1 OR '1' = '1' AND pass ='1' OR '1' = '1'"
```

음.. 깔끔한가요? 아닌가요? 깔끔하게 만들어보죠...

여러분이 실제 테이블을 안다면, 이해하는데 아무 문제 없을겁니다.

1 OR 1 = 1 ??

1 OR 1은 1과 같은가요? 확실히 맞~죠!!!

그러면, 'sql_inj' 테이블에서 누가 첫번째 사용자죠?
아마. 웹어플리케이션을 설치한 사람이겠죠? 그렇죠! 관리자에요 :D

그래서, 첫번째 유저로서 로그인한거고, 첫번째 유저는 관리자...그러면
우리는 관리자. ;) 확실히 맞죠?

또한 다음 쿼리 스트링으로도 넣을 수 있습니다:

```
$nick = 1' OR '1' = '1' #
$pass = what_we_want_to_put  // 우리가 넣길 원하는 것
```

새로운 쿼리는 다음과 같을 거에요:

```
"SELECT * FROM sql_inj WHERE nick ='1 OR '1' = '1' # AND pass = what_we_want_to_put"
```

앞서 말씀드린 것처럼 '#'은 후에 주석이라는 것을 의미하죠! 최신 :D 쿼리는 :

"SELECT * FROM sql_inj WHERE nick ='1 OR '1' = '1'

그리고... 우리는 다시 관리자가 되는 거에요 :D

------[0x04b: 쿼리가 하나? 아니죠.. 2개에요!]

우리는 SQL 인젝션으로 쿼리를 수정할 수 있고 새로운 데이터를 삽입하고, 사용자 정보를 변경하며, 더 많은 일들을 할 수 있습니다. 아래 소스코드를 봅시다 :

```
<?php

//email.php

[...]

$email = $_POST['email'];

[...]

$query = mysql_query("SELECT email, passwd, user_name FROM users WHERE email =
'".$email."'");
[...]

?>
```

오 오 오.. 여기서 무엇을 얻나요? $email 변수는 사용되기 전에 적절하게 처리되어 있지 않습니다. 좋아요!! 우리는 이걸 익스플로잇 할 수 있고 그리고...예를 들어 아래와 같이 변수로의 삽입을 통해서 데이터베이스 정보를 쉽게 변경합니다.

```
$email =  x'; UPDATE users SET email = 'omnipresent@email.it'
           WHERE email = 'admin@site.com ';
```

새로운 쿼리는 다음과 같을 거에요:

```
SELECT email, passwd, user_name FROM users
         WHERE email = ' x'; UPDATE users SET email = 'omnipresent@email.it'
         WHERE email = 'admin@site.com ';
```

여기서 공격자는 'users' 테이블을 변경시켰고, 특별히 관리자의 e-mail 필드를 공격자의 e-mail로 변경시켰습니다. 그렇기 때문에 , 예를들어 "패스워드를 잊어버리셨습니까?"와 같은 문장을 공격자의 e-mail을 통해서 보낼 수 있습니다.(omnipresent@email.it이죠 :D) 그리고 다음과 같은 E-mail을 받습니다 :

```
         From: host@site.com
         To: omnipresent@email.it
         Subject: Login Password

         이예.. 받아요 ;)

         Username: Admin
         Password: 12345

         관리자로
         여기다.
```

---[Patching]---

가장 먼저, 스크립트에 대해서 패치하기 위해 php.ini파일에 근사한 수정을 선택할 수

있습니다. :

1. magic_quotes_gcp를 On으로 설정
  magic_quotes_gcp는 정점(')전에 에스케이프 문자를 넣을 것입니다.
        -COOKIE
        -POST
        -GET

2. addslashes()를 사용한다
   addslashes()는 '/'와 함께 인용할 것입니다.

3. htmlspecialchars()
   htmlspecialchars()는 HTML 코드안의 특수한 문자를 변환시킬 겁니다.

4. mysql_escape_string()
   mysql_query에 스트링이 사용되는 것을 벗어납니다.

5. 좀 더 많은 기능을 위해 www.php.net을 보세요 ;)

6. 아래와 같이, 사용자로부터의 입력을 확인하세요.

         $user_id = (int)$_GET['user_id'];

         $user_id는 언제나 정수이고 안전한 웹 어플리케이션을 위해 변환(cast)
        할 수 있습니다.

---[/]

---[0x05: File Traverse]

파일시스템을 가르지르는 것은 정말로 중요하고 위험한 버그입니다; 그리고 이 버그의
적은 설명은 제가 할수 있는 만큼 최소로 하겠습니다. :D

우리가 파일을 어디서 이용하던지, 스크립트의 특정 지점에는 파일이름과 파일경로를
지정해야합니다. 많은 경우에, 파일이름은 인자(argument)로서 사용자에 의해 전달됩
니다; 인자(argument)는 fopen()과 같은 함수로 전달됩니다:

```
<?php

[...]

         $fp = fopen("/path/{$_GET['filename']}.txt", 'r');

[...]

?>
```

이 스크립트에는 'filename'이 원격 사용자에 의해 조작될 수 있는 취약점이 존재합니다.
이러한 경우에 공격자는 "../"와 같이 상위 디렉토리로 이동해서 다른 파일들을 볼 수
있는 복합적인 것들을 이용해 파일시스템을 가로지를 수 있습니다.

예를 들어:

```
http://remote_host/path/bug.php?filename=../../../../path_of_another_file/file%00
```

그렇지만, NULL byte(%00)은 파일이름 확장자 제한을 피하고 스트링을 종료하기 위해
많은 공격에 사용된다는 것을 유념하세요.

---[Patching]---

이 스크립트를 패치하기 위한 좋은 해결책은 다음 예제와 같이 basename() 함수를 사용하는 것입니다 :

```
<?php

          $clean = array();

          if (basename($_GET['filename']) == $_GET['filename'])
                    {
                              $clean['filename'] = $_GET['filename'];
                    }
          else
                    {

                              [...]

                    }

$fp = fopen("/path/{$clean['filename']}.txt", 'r');


?>
```

----------[/]

---[0x06: Conclusion]

이제 저의 문서 "PHP Underground Security" 의 마지막이네요. 이 문서가 여러분이 안전한 PHP 코드를 만드는데 도움이 되었으면 합니다!

어떤 문제가 있다면 다음 메일로 보내주세요.
omnipresent@email.it or omni@playhack.net.

----------[/]

W==========[EOF]==========/

milw0rm.com [2007-04-12]