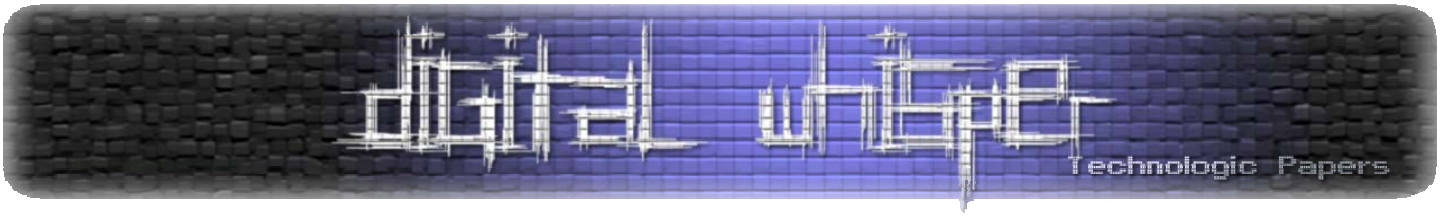


פרצות אבטחה נפוצות בהעלאת קבצים בעזרת PHP

מאת Hyp3rInj3cT10n

העלאת קבצים ושיתופם הוא עניין שהפך לנפוץ מאוד בימינו: לא פעם ולא פעמיים אנחנו נתקלים במצבים שבהם אנו צריכים ו/או רוצים לשתף קבצים - בין אם מדובר בתמונות, סירטונים או קבצים מסוגים אחרים שנועדו בין השאר לצפיה או להורדה. מפתחי ה-PHP יצרו עבור בונה האתר מספר פונקציות כדוגמת הפונקציה move_uploaded_file, שמעלה קבצים לשרת. הפונקציה עושה את עבודתה בצורה יפה מאוד, אך משאירה את האחריות לבדוק את הקובץ בידי המתכנת עצמו. נתמקד במאמר זה בפרצות נפוצות ואפשריות שנוצרות בעת תיכנות מערכות העלאת קבצים.

לשם הנוחות, נשתמש בדוגמאות בשפת PHP אך רב הדוגמאות שאביא מקבילות גם בשפות צד שרת אחרות.

אפשרויות בקובץ ההגדרות של ה-PHP

לפני שנתחיל להתעסק בסקריפטים ב-PHP, אנחנו צריכים להכיר ולטפל קודם כל בסביבת העבודה שלנו: השרת. ישנן מספר הגדרות שנרצה לערוך ולוודא. ההגדרה הראשונה: בקובץ ההגדרות של PHP קיימת ההגדרה file_uploads, שקובעת האם יהיה ניתן להעלות קבצים באמצעות פרוטוקול ה-HTTP. אנחנו כמובן נוודא שאפשר להעלות קבצים באמצעות ה-HTTP, נוודא שההגדרה אכן דלוקה, כלומר:

```
file_uploads = On
```

upload_max_filesize קובעת את הגודל המירבי המותר לקובץ המועלה באמצעות ה-HTTP. עם זאת, אל תבנו על ההגדרה הזו לצורך אבטחת השרת שלכם – תמיד עדיף לבצע בדיקה ידנית שאותה נציג. לצורך הדוגמא נגדיר:

```
upload_max_filesize = 10M
```

מאחר ואנו משתמשים במתודת POST לצורך עניין העלאת קבצים יש עוד מספר דברים להגדיר. ההגדרה post_max_size מייצגת את הגודל המירבי של בקשה המשתמשת בשיטה POST. הגדרה זו משפיעה גם על העלאות קבצים באמצעות השיטה POST, ולכן יהיה כדאי להשאיר את הערך של ההגדרה הזו בדוגמא לערך של ההגדרה upload_max_filesize או ערך הגדול ממנו כדי למנוע בעיות עם גודלו המירבי של הקובץ. בדוגמאות במסמך זה נשתמש בהגדרה הבאה:

```
post_max_filesize = 11M
```

max_input_time מגדירה את הזמן (בשניות) שמותר ל-PHP לחכות כדי לקבל את המידע. בעת העלאת קבצים גדולים בחיבורי אינטרנט איטיים, הזמן עלול לעבור את הערך max_input_time, ואז ההעלאה תפסק ותכשל. אפשר להגדיר את max_input_time כמספר גדול כדי למנוע מצב כזה:

```
max_input_time = 9999
```

עם זאת, עדיף להשאיר הגדרה זו בערך מעשי, למשל 30 או 60 כדי למנוע ניצול משאבים גרוע.

```
max_input_time = 60
```

MAX_FILE_SIZE מגדיר את גודל הקובץ המקסימלי, האומנם?

להלן דוגמא שכיחה לקוד לבדיקת גודל הקובץ:

```
<?php
if ( isset($_FILES['file']) )
{
    $name = $_FILES['file']['name'];
    $tmp = $_FILES['file']['tmp_name'];
    $error = $_FILES['file']['error'];

    if ( $error == UPLOAD_ERR_OK && move_uploaded_file($tmp, $name) )
    {
        echo 'Your file has been uploaded.';
    }
    else
    {
        echo 'An error has been occurred, file not uploaded.';
    }

    echo '<br /><br />';
}

echo <<<END
<form action="?" method="POST" enctype="multipart/form-data">
    File: <input type="file" name="file" />
    <input type="hidden" name="MAX_FILE_SIZE" value="100000" />
    <input type="submit" value="Upload" />
</form>
END;
?>
```

הגדרנו בעזרת MAX_FILE_SIZE את גודל הקובץ המקסימלי ל-100 אלף בייטים. אם גודל הקובץ יעלה על הגודל שצויין תוחזר שגיאה והתהליך יפסק. בפתרון זה יש בעיה גדולה: MAX_FILE_SIZE מוגדר בצד הלקוח. כלומר, הוא ניתן לשינוי. אפשר לשנות את MAX_FILE_SIZE באמצעות Javascript Injection, Form Manipulation ודרכים רבות נוספות.

דוגמא ל-Form Manipulation

להלן הטופס שלנו:

```
<form action="?" method="POST" enctype="multipart/form-data">
    File: <input type="file" name="file" />
    <input type="hidden" name="MAX_FILE_SIZE" value="100000" />
    <input type="submit" value="Upload" />
</form>
```

כפי שניתן לראות בבירור, MAX_FILE_SIZE מוגדר בטופס שלנו כ-100,000 בתוך שדה input מוסתר (hidden). האפשרות הפשוטה ביותר היא להשתמש באחת מתוך אלפי התוספות של הדפדפן Firefox בשביל לשנות את סוג ה-input ממוסתר לטקסט (text), כדי שנוכל לשנות את ה-100,000 לכל ערך שנבחר, בהתאם למה שנרצה להעלות.

אפשרות שניה שחשוב להכירה היא השיטה הידנית - שיכתוב הקוד ללא כלים: נעתיק את הקוד ונשנה את שדה ה-hidden ל-text, (נוכל גם לשנות את ערך ברירת המחדל אם נרצה). בסיום העריכה נשמור את הקובץ כ-HTML. הקוד שלנו צריך להראות כך:

```
<form action="?" method="POST" enctype="multipart/form-data">
    File: <input type="file" name="file" />
    <input type="text" name="MAX_FILE_SIZE" value="5000000" />
    <input type="submit" value="Upload" />
</form>
```

כדי שהדוגמא תעבוד יש לעדכן את כתובת הטופס (דבר שאנשים נוטים לשכוח). נחזור שוב לשורה הראשונה:

```
<form action="?" method="POST" enctype="multipart/form-data">
```

הסימן **?** (סימן שאלה) מייצג למעשה את שורת הכתובת, והמשתנים שבאים לאחריה. זוהי דרך קצרה מאוד להפנות את הקובץ לעצמו, במקום להשתמש במשתנים ולהסתבך בחיפוש שם וכתובת הקובץ. כדי שהטופס יעבוד נשנה את הערך של ה-action לכתובת של הקובץ המעלה את הקבצים, לדוגמא:

```
<form action="http://example.com/upload.php" method="POST"
enctype="multipart/form-data">
```

שימו לב שבמידה ויש מידע המקשר אל עמוד ההעלאה והוא מועבר באמצעות שורת הכתובת יש לצרף אותו. לדוגמא:

```
<form action="http://example.com/index.php?page=upload&terms=agreed"
method="POST" enctype="multipart/form-data">
```

דוגמא ל-Javascript Injection

דרך נוספת היא שימוש בקוד Javascript כדי לשנות את הערך של MAX_FILE_SIZE. שיטה זו נקראית לעתים גם **Javascript Manipulation**. אנחנו משתמשים בכך שניתן לכתוב פקודות JavaScript בשורת הכתובת של הדפדפן, שיפעלו על החלון הנוכחי. נסו למשל לכתוב בדפדפן שלכם בשורת הכתובת את השורה:

```
javascript:alert('Can you see this?');
```

הריצו את הקוד בדפדפן שלכם, בשורת הכתובת. תקפוץ לכם הודעת Alert עם הטקסט.

כעת נראה את אופן הניצול האפשרי בדוגמא שהצגנו בפרק הקודם. הפקודה הבאה תראה את הערך של MAX_FILE_SIZE:

```
javascript:alert(document.getElementsByName("MAX_FILE_SIZE")[0].value);
```

הפקודה הבאה תשנה ערך זה:

```
javascript:document.getElementsByName("MAX_FILE_SIZE")[0].value=999999;return;
```

השתמשנו ב-getElementsByName, ובחרנו ב-0 שמייצג את המופע הראשון (והיחיד). הגדרנו מחדש את ערכו של MAX_FILE_SIZE כ-999999.

צורת כתיבה נוספת:

```
javascript:void(document.getElementsByName("MAX_FILE_SIZE")[0].value=999944);
```

אם לא נשתמש ב-void (או ב-return) אנחנו נועבר לדף שאינו קיים או לדף לבן - הדפדפן רוצה להציג לנו את הפלט של הקוד שלנו. לקוד שאנחנו רוצים לבצע לא אמור להיות פלט מיוחד ואנחנו רוצים להשאר באותו הדף, ולשם כך בדיוק אנו משתמשים ב-void. פונקציה זו לא מחזירה פלט, ובעזרתה אנחנו לא נועבר לדף לא רצוי.

לפני שנמשיך, נראה יתרון משמעותי נוסף שנותנת לנו הפונקציה alert, בעזרת הדוגמא הבאה:

```
javascript:alert(document.getElementsByName("MAX_FILE_SIZE")[0].value);
```

לא עשינו שום דבר מיוחד – הכנסנו את הערך של MAX_FILE_SIZE לפונקצית ה-Alert. לפעולה הזאת יכולות להיות 2 תגובות אפשריות:

1. תקפוץ הודעת Alert עם התוכן של MAX_FILE_SIZE. נוכל לראות שבאמת שינינו את ערכו.
2. לא יקרה כלום. נסיק כי טעינו והפניה לא נכונה ו/או שהערך המוחזר הוא ריק. (יש דפדפנים שכן יציגו Alert ריק)

נמשיך לעוד דוגמא שבעזרתה נוכל לערוך את הערך ב-MAX_FILE_SIZE:

```
javascript:void(document.getElementsByName("MAX_FILE_SIZE")[0].type='text');
```

שורה זו משנה את הסוג של ה-input מ-hidden לטקסט. כעת, ניתן לשנות את הערך שלו בטופס עצמו.

התגוננות

כדי להתגונן נפעיל בדיקה נוספת בצד השרת לבדיקת הערך של MAX_FILE_SIZE:

```
<?php
$maxFileSize = 100000;
if ( isset($_FILES['file']) )
{
    $name = $_FILES['file']['name'];
    $tmp = $_FILES['file']['tmp_name'];
    $error = $_FILES['file']['error'];
    if ( isset($_POST['MAX_FILE_SIZE']) &&
              $maxFileSize == $_POST['MAX_FILE_SIZE'] )
    {
        if ( $error == UPLOAD_ERR_OK && move_uploaded_file($tmp, $name) )
        {
            echo 'Your file has been uploaded.';
        }
        else
        {
            echo 'An error has been occurred, file not uploaded.';
        }
    }
    else
    {
        echo 'Were you trying to trick us?';
    }

    echo '<br /><br />';
}
echo <<<END
<form action="?" method="POST" enctype="multipart/form-data">
    File: <input type="file" name="file" />
    <input type="hidden" name="MAX_FILE_SIZE" value="{$maxFileSize}" />
    <input type="submit" value="Upload" />
</form>
END;
?>
```

נוכל גם לכתוב פתרון שלא יתבסס כלל על MAX_FILE_SIZE:

```
<?php
if ( isset($_FILES['file']) )
{
    $name = $_FILES['file']['name'];
    $tmp = $_FILES['file']['tmp_name'];
    $error = $_FILES['file']['error'];
    $size = $_FILES['file']['size'];
    if ( $size < 100000 )
    {
        if ( $error == UPLOAD_ERR_OK && move_uploaded_file($tmp, $name) )
            echo 'Your file has been uploaded.';
        else
            echo 'An error has been occurred, file not uploaded.';
    }
    else
    {
        echo 'Were you trying to trick us?';
    }
    echo '<br /><br />';
}
echo <<<END
<form action="?" method="POST" enctype="multipart/form-data">
    File: <input type="file" name="file" />
    <input type="submit" value="Upload" />
</form>
END;
?>
```

החלפת קובץ שמועלה בקובץ קיים

מה יקרה אם נרצה להעלות תמונה בשם bg.jpg בזמן שהיא כבר קיימת על השרת? התמונה החדשה שנעלה תחליף את הישנה. מה יקרה אם תוקף יעלה קובץ שתוכנו "Hacked" בשם זהה לזה של קובץ האינדקס של המערכת? יכולים להווצר מצבים בהם קבצים חשובים באתר מוחלפים בקבצים אחרים!

הגדרת מיקום הקובץ

מומלץ מאוד להגדיר תיקיה נפרדת לקבצים שמועלים לשרת, כדי למנוע מצבים בהם מוחלף קובץ קיים באחר. השורה בדוגמא שמגדירה את האופן שבו ישמר הקובץ היא:

```
$name = $_FILES['file']['name'];
```

כדי להכניס את הקובץ לתיקיה, פשוט נוסיף את שמה:

```
$name = 'uploads/'.$_FILES['file']['name'];
```

כעת, הקבצים יועלו לתיקיה uploads.

הגדרת שם הקובץ

הצלחנו לגרום לקבצים לא להחליף את הקבצים שבתיקייה שלנו בעזרת הפרדה שלהם לתיקיה אחרת, אך עדיין לא פתרנו את כל הבעיות. לדוגמא, מה יקרה אם נעלה קובץ בשם program.exe שסוגר פירצות אבטחה במחשב, ולאחר מכן גולש אחר יעלה וירוס קטלני בשם program.exe? הקובץ החדש יחליף את הקובץ הקיים, ויהיה קשה לנו להבחין בכך. הפתרון שלנו יהיה כמובן בעזרת שינוי שמות הקבצים. נגדיר שמות חדשים לקבצים שמועלים: ניצור לקבצים שמועלים על ידי הגולשים שמות חדשים משלנו. נדאג ששמות שניצור לקבצים שמועלים לא יוכלו לחזור על עצמו. לדוגמא, פתרון אפשרי יכול להיות:

```
$name = 'uploads/'.time().crc32($_FILES['file']['name'].time()).'-
'.$_FILES['file']['name'];
```

הוספנו לשם הקובץ מקדם שסביר להניח שלא יוכל לחזור על עצמו, מאחר והוא מתבסס על שני מרכיבים דינאמיים מאוד: שם הקובץ והזמן הנוכחי. ההסתברות ששני קבצים עם שם זהה יועלו במקביל היא אפסית.

סיומות קבצים

הגנה נוספת היא להגביל את סוג הקבצים שניתן להעלות לשרת (סוג הקבצים – על פי הסיומת שלהם). בדרך כלל אנשים נוטים להגדיר סיומות אסורות (Black List Filter), ולא אילו סיומות מותרות (White List Filter). למרות שזו גישה מאוד אינטואיטיבית, היא אינה נכונה ברוב המקרים – נראה זאת כעת. להלן דוגמת קוד לבדיקת סיומות אסורות:

```
<?php
$maxFileSize = 100000;
$exts = array('php','cgi','html');
if ( isset($_FILES['file']) )
{
    $name = 'uploads/'.time().crc32($_FILES['file']['name'].time()) .'-
'.$_FILES['file']['name'];
    $tmp = $_FILES['file']['tmp_name'];
    $error = $_FILES['file']['error'];

    if ( isset($_POST['MAX_FILE_SIZE']) && $maxFileSize ==
$_POST['MAX_FILE_SIZE'] )
    {
        $extension = pathinfo($tmp);
        $extension = strtolower($extension['extension']);

        if ( in_array($ext,$exts) )
        {
            echo 'Bad extension';
        }
        else
        {
            if ( $error == UPLOAD_ERR_OK && move_uploaded_file($tmp, $name) )
```

```
            {
                echo 'Your file has been uploaded.';
            }
            else
            {
                echo 'An error has been occurred, file not uploaded.';
            }
        }
    }
    else
    {
        echo 'Were you trying to trick us?';
    }
    echo '<br /><br />';
}
$extsText = implode(', ',$exts);

echo <<<END
<form action="?" method="POST" enctype="multipart/form-data">
    File: <input type="file" name="file" />
    <input type="hidden" name="MAX_FILE_SIZE" value=" {$maxFileSize}" />
    <input type="submit" value="Upload" /><br />
    Disabled Extensions: {$extsText}.
</form>
END;
?>
```

אנחנו מקבלים רשימה של סיומות קבצים אסורות. אם המערכת מזהה סיומת מסוכנת, תהליך ההעלאת הקובץ מבוטל. נעבור כעת על מספר דוגמאות לדרכי ניצול פוטנציאליות במערכות הבנויות באופן זה או דומה.

הרצת קוד בצד השרת (למשל PHP)

בדוגמא אמנם הסיומת php חסומה, אך עדיין עומדות לרשותינו עוד הרבה סיומות שגם הן מריצות קודים ב-PHP, למשל: php3, php4, php5, php6, phtml. יש לא מעט אנשים שמעולם לא נתקלו בסיומות האלה ולכן לא יחסמו אותן כאשר הם ישבו לכתוב את הפילטר שהצגנו קודם לכן. בנוסף שימו לב שיש אפשרות לנסות להריץ קבצי perl ו-shtml אם השרת תומך בכך.

הרצת קודים עם שפות צד לקוח

הסיומת html אומנם חסומה לנו, אך בכל זאת נוכל לנסות להריץ קודים ב-HTML על ידי שימוש בסיומת htm. מאחר ואנו יכולים להריץ קודים ב-HTML, נוכל גם לנסות להפעיל ולהריץ Javascript, Flash, Java VBScript ועוד...

htaccess ו-htpasswd

קבצי htaccess ו-htpasswd ועניני אבטחה הקשורים בהם מופיעים במאמר מאת אפיק בגליון זה. קבצים אלו נשמרים בשמות: htaccess., htpasswd. שימו לב - שמם של הקבצים הוא בעצם סיומת בלבד! בעלי אתרים רבים מכירים את הסיומות האלה, אבל הם לא חושבים עליהם כשהם חוסמים סיומות קבצים.

כדי לספק לכם המחשה לסכנה, בואו נראה מספר דוגמאות מסוכנות לפעולות הניתנות לביצוע בעזרת קובץ ה-htaccess:

- חסימת הכניסה לתיקיה: (כך אף אחד לא יכול להכנס לתיקיה בעזרת דפדפן האינטרנט)

```
deny from all
```

- שינוי שם קובץ האינדקס לקובץ שלנו: (בהנחה שהעלנו קובץ הנקרא deface.html)

```
DirectoryIndex deface.html
```

- הפניית הגולש מקובץ האינדקס לקובץ מרוחק:

```
Redirect index.php http://my-site.com/deface.html
```

- סגירת התיקיה בסיסמה:

```
AuthName "Please Identify"
AuthType Basic
AuthUserFile .htpasswd
Require valid-user
```

קוד זה יקפיץ חלון המבקש שם משתמש וסיסמה, כאשר הכותרת תהיה Please Identify.

במקרה הזה נצטרך גם להשתמש ב-htpasswd. קובץ זה יכיל שמות משתמשים וסיסמאות, לדוגמא:

```
username:encrypted-password
username:encrypted-password
username:encrypted-password
```

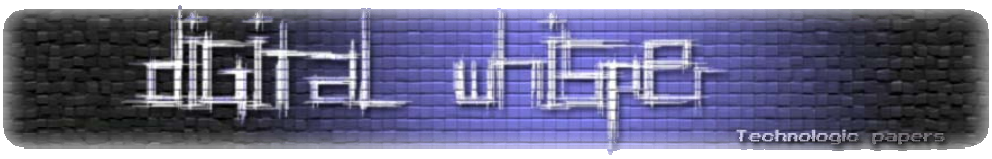
שם המשתמש מופרד מהסיסמה המוצפנת בעזרת התו : (נקודתיים), וכל שורה מפרידה בין המשתמשים האחרים. על אופן הצפנת הסיסמה הרחבנו במאמר אחר בגליון זה. בנוסף יש מספיק כלים באינטרנט לטיפול בעניין, כמו:

http://tools.dynamicdrive.com/password/

התגוננות

נגדיר רשימה של סיומות מותרות, במקום רשימה של סיומות אסורות:

```
<?php
$maxFileSize = 100000;
$exts = array('jpg','bmp','png','gif','txt','rar','doc','ppt'); //etc...
if ( isset($_FILES['file']) )
{
    $name = 'uploads/'.time().crc32($_FILES['file']['name'].time()) .'-
'.$_FILES['file']['name'];
    $tmp = $_FILES['file']['tmp_name'];
    $error = $_FILES['file']['error'];
    if ( isset($_POST['MAX_FILE_SIZE']) && $maxFileSize ==
                    $_POST['MAX_FILE_SIZE'] )
    {
        $extension = pathinfo($tmp); $extension =
strtolower($extension['extension']);
        if ( !in_array($ext,$exts) )
            echo 'Bad extension';
        else
            if ( $error == UPLOAD_ERR_OK && move_uploaded_file($tmp, $name) )
                echo 'Your file has been uploaded.';
            else
                echo 'An error has been occurred, file not uploaded.';
    }
    else echo 'Were you trying to trick us?';
    echo '<br /><br />';
}
$extsText = implode(', ',$exts);
echo <<<END
<form action="?" method="POST" enctype="multipart/form-data">
    File: <input type="file" name="file" />
    <input type="hidden" name="MAX_FILE_SIZE" value=" {$maxFileSize}" />
    <input type="submit" value="Upload" /><br />
    Allowed Extensions: {$extsText}.
</form>
END;
?>
```



Null Byte Poisoning

ה-Null Byte הוא התו הראשון בטבלת ה-ASCII, ובדרך כלל המערכת נעזרת בו בכדי לזיהות מחרוזות. ב-HEX התו מיוצג כ-00 (או 00%) וב-PHP:

```
chr(0)
```

ה-Null Byte הוא תו בלתי נראה. לא ניתן לראות אותו.

Null Byte Poisoning (מוכר גם כ-Null Byte Attack)

לצורך ההסבר, נגיד שהתו & (אמפטסנד) יהיה ה-Null Byte, כדי שאוכל להעביר בצורה טובה יותר את ההדגמה. דמיינו לכם מערכת הכוללת פונקציה להעלאת קבצים כמו זו שנמצאת בדוגמת הקוד האחרונה עם הסיומות. כעת בואו נראה מה למשל יקרה אם שם הקובץ שנכניס יהיה השם הבא: image.jpg**&.php** בבדיקה שנעשה בעזרת ה-PHP, הביטוי שיבדק יהיה אך ורק: image.jpg והבדיקה תעבור בהצלחה. כאשר הקובץ יווצר, ה-Null Byte יצונזר ושם הקובץ יהיה: image.jpg.php

התגוננות

לכאורה נראה כי מדובר בשיטה שלא ניתנת לעצירה, אך זה לא נכון. כדי להתגונן יש להבריח או לצנזר את ה-Null Byte כך שלא יוכל לתפקד. נוכל לצנזר את ה-Null Byte בעזרת השורה הבאה:

```
$name = str_replace(chr(0),'',$_FILES['file']['name']);
```

ה-Magic Quotes מבצעים הברחה אוטומטית ל-Null Byte, אך כמובן שלא באמת נסמוך עליהם.

Local File Disclosure

בכל הדוגמאות עד כה שהצגנו, הצגנו הודעה אם הקובץ הועלה או שהתרחשה שגיאה והוא לא הועלה. לדוגמאות זה היה נחמד מאוד, אבל כשאנחנו מדברים על מערכת אמיתית, אנחנו נספק למשתמש קישור להורדת הקובץ. אין בעיה לתת קישור כזה - הרי יש לנו את המשתנה name שמכיל את שם הקובץ, ונוכל להשתמש בו כהפניה. צריך ליצור קובץ שיודיע לדפדפן להוריד את הקובץ, לדוגמא:

```
<?php
if ( isset($_GET['file']) && is_string($_GET['file']) )
    if ( @file_exists($_GET['file']) && @is_readable($_GET['file']) )
    {
        header("Content-Type: application/octet-stream");
        header('Content-Disposition: attachment;
filename="'.$_GET['file'].'"');
        readfile('uploads/'.$_GET['file']);
    }
?>
```

שורת הכתובת צריכה להיות כזאת: (כאשר filename מייצג את שם הקובץ שנרצה להוריד)

```
?file=[filename]
```

Path Traversal

השיטה Path Traversal (מוכרת גם כ-Directory Traversal) משמשת אותנו ל"חציית" התיקיה בה אנו נמצאים. יש 2 סימונים מוכרים שיש להכיר:

- **. (נקודה אחת)** - מייצגת את התיקיה הנוכחית
- **.. (שתי נקודות)** - מייצגות תיקייה אחת למעלה. (Parent Directory)

מה אם נשתמש ב-Path Traversal כדי לעלות תיקיה למעלה?

```
?file=../index.php
```

מה שיקרה זה:

```
readfile('uploads/../index.php');
```

כלומר, ניגש לתיקיה uploads ואז לתיקיה מעליה (חזרנו אחורה), ומשם נקרא את הקובץ שנקרא index.php. המשמעות היא שכעת נוכל להוריד ולקרוא כל קובץ שנרצה מהאתר.

בתוך קובץ האינדקס נוכל למצוא בדרך כלל פקודות ייבוא לקבצי תצורה אחרים, שגם אותם נוכל להוריד למחשב ולקרוא. לדוגמא, אפשר למצוא בדרך כלל בקבצי האינדקס של המערכת שמות/מיקום של קבצי הקונפיגורציה הכוללים מידע שימושי רב, כמו למשל את פרטי ההתחברות למסד הנתונים (Connection String) המשמש את המערכת באיכסון המידע, קריאות לקבצי קונפיגורציה התומכים במערכת וכו'. אם זה לא מספיק, תמיד אפשר לחפש אחר קבצי htaccess, htpasswd ועוד קבצים שאין גישה אליהם דרך המערכת.

התגוננות

ההתגוננות היא פשוטה מאוד: נצנזר את התווים .. (שתי נקודות) ו-/ (סלאש). לדוגמא:

```
$file = $_GET['file'];
$file = str_replace('/','',$file);
$file = str_replace('..','',$file);
```

יש צורך להחליף אף את התווים הנ"ל בסוגי הקידודים השונים הנתמכים ע"י טכנולוגיית המערכת.

File Download Injection

להלן הדוגמא שבה נשתמש:

```
<?php
if ( isset($_GET['file']) && is_string($_GET['file']) )
{
    $file = $_GET['file'];
    $file = str_replace(chr(0),'',$file);
    $file = str_replace('..','',$file);
    header("Content-Type: application/octet-stream");
    header('Content-Disposition: attachment; filename="'.$file.'"');
    readfile('uploads/'.$file);
}
?>
```

התגובה שנקבל מהשרת כשניגש לקובץ תהיה בערך כזאת:

```
HTTP/1.1 200 OK
Date: ...
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=[file name]
Content-Length: [file length]
[file content]
```

כשנגש לקובץ בצורה הבאה:

```
?file=roy.bat%0a%0dContent-Length%3A%2016%0a%0d%0a%0dshutdown%20-s%20-t%2060
```

- CRLF = a%0d (ירידת שורה)
- A%3 = : (נקודתיים)
- %20 = רווח

כלומר:

```
roy.bat
Content-Length: 16

shutdown -s -t 60
```

נקבל:

```
HTTP/1.1 200 OK
Date: [...]
Content-Type: application/octet-stream
Content-Disposition: attachment; filename=roy.bat
Content-Length: 16

shutdown -s -t 60
Content-Length: [...]
```

הדפדפן יתייחס לכותר ה-Content-Length הראשון שישלח לו מהשרת, ולכן יקרא רק את 16 התווים הראשונים. למעשה, הדפדפן יוריד קובץ שנקרא roy.bat ותוכנו:

```
shutdown -s -t 60
```

הקובץ הוא קובץ אצווה וברגע שהגולש יפתח את אותו, תופיע לו הודעה שמחשבו ייכבה בעוד 60 שניות. אתם מוזמנים להפעיל את זה כדי לבדוק. לביטול הפעולה יש להריץ בשורת ההפעלה או ב-cmd את זה:

```
shutdown -a
```

במקרה הזה השתמשתי דווא בקובץ אצווה (batch), אך כמובן שנוכל להשתמש במגוון רחב של קבצים.

נוודא כי הקובץ אכן קיים וניתן לקריאה לפני שניתן אותו לגולש. לדוגמא:

```
<?php
if ( isset($_GET['file']) && is_string($_GET['file']) )
{
    $file = $_GET['file'];
    $file = str_replace('/','',$file);
    $file = str_replace('..','',$file);

    if ( @is_readable($_GET['file']) )
    {
        header("Content-Type: application/octet-stream");
        header('Content-Disposition: attachment; filename="'.$file.'"');
        readfile('uploads/'.$file);
    }
}
?>
```

עקיפת מכסת הקבצים שניתן להעלות במקביל (לביצוע DoS)

לפעמים נרצה לאפשר העלאה של יותר מקובץ אחד במקביל, למשל 5 קבצים. כך יראה הטופס:

```
<form action="?" method="POST" enctype="multipart/form-data">
    File #1: <input type="file" name="file1" /><br />
    File #2: <input type="file" name="file2" /><br />
    File #3: <input type="file" name="file3" /><br />
    File #4: <input type="file" name="file4" /><br />
    File #5: <input type="file" name="file5" /><br />
...
```

לשם ביצוע ההתקפה יש לנו מספר אפשרויות:

1. בניית פונקציה שמתעסקת בהעלאת הקובץ, ונזמן אותה 5 פעמים.
2. העתקת הקוד והדבקתו עוד 4 פעמים, כשבכל פעם נשנה את שם המשתנה שאיתו עובדים.
3. להשתמש בלולאת foreach.

הפיתרון השלישי הוא הנפוץ ביותר כיום והוא גם הפתרון הנוח ביותר לשימוש, אך הוא גם הפתרון הלוקה בחסר.

דוגמא לקוד המבצע את הפיתרון השלישי:

```
foreach ( $_FILES as $file )
{
    $name = 'uploads/'.time().crc32($file['name'].time()).'-'.$file['name'];
    $tmp = $file['tmp_name'];
    $error = $file['error'];
    if ( isset($_POST['MAX_FILE_SIZE']) && $maxFileSize ==
$_POST['MAX_FILE_SIZE'] )
    {
        $extension = pathinfo($tmp); $extension =
strtolower($extension['extension']);
        if ( !in_array($ext,$exts) )
            echo 'Bad extension';
        else
            if ( $error == UPLOAD_ERR_OK && move_uploaded_file($tmp, $name) )
                echo 'Your file has been uploaded.';
            else
                echo 'An error has been occurred, file not uploaded.';
    }
    else echo 'Were you trying to trick us?';
    echo '<br /><br />';
}
```

הפיתרונות הראשון והשני הם סטאטים, ומיועדים להעלות רק עד חמישה קבצים בכל מילוי טופס שמתבצע, ולא יותר. לעומתם, הפיתרון השלישי יכול להעלות כמות משתנה של קבצים- כמספר הקבצים שהוא מקבל. לכן, אופן הניצול יהיה Form Manipulation - נערוך את הטופס ונשמור בקובץ חדש: (לא לשכוח את ה-action!)

```
<form action="[file-url]" method="POST" enctype="multipart/form-data">
    File #1: <input type="file" name="file1" /><br />
    File #2: <input type="file" name="file2" /><br />
    File #3: <input type="file" name="file3" /><br />
    File #4: <input type="file" name="file4" /><br />
    File #5: <input type="file" name="file5" /><br />
    File #6: <input type="file" name="file6" /><br />
    File #7: <input type="file" name="file7" /><br />
    File #8: <input type="file" name="file8" /><br />
    File #9: <input type="file" name="file9" /><br />
    File #10: <input type="file" name="file10" /><br />
...
```

התגוננות

לפתירת הבעיה קיימות מספר אפשרויות, אך מספיק להציג שתיים הן: שבירת הלולאה וביטול הלולאה.

דוגמא ראשונה - שבירת הלולאה:

```
foreach ($_FILES as $key => $array)
{
       if ( $key == 4 ) // 0,1,2,3,4 = 5
              break;
       …
}
```

דוגמא שניה - ביטול הלולאה:

```
if ( count($_FILES) <= 5 )
{
    foreach ($_FILES as $array)
```

סיכום

במאמר זה נגעתי במספר נרחב של נקודות המופיעות בהרבה מנגנוני העלאת קבצים במערכות השונות. חשוב לזכור שבכל מערכת ומערכת יכולים להווצר סוגים שונים של חורים, אך אחד העקרונות החשובים ביותר כשמדובר בפיתוח במערכות המקבלות קלט (כל קלט שהוא) מהמשתמש הוא שלעולם אין לסמוך על המשתמש ותמיד יש לבצע בדיקות מקיפות ולוודא שאכן הקלט עומד בסטנדרטים שקבענו.