Reference: OWASP best practices <https://www.owasp.org/index.php/Unrestricted_File_Upload>

### INPUT

1. The file upload function must only be accessible to authenticated and authorized users.
2. The upload service must whitelist allowed file types and limit to minimum necessary to support business requirements.
   * Recommendation
     + Image file types: .jpg .png
     + Document file types: .pdf .pdfx .doc .docx
   * This list determines the types of files that can be uploaded, and rejects all files that do not match approved types.
   * If user uploads an invalid file type (.exe) but names it is named a valid file type (.pdf) then application should prevent.
   * Do not allow compressed files to be uploaded
3. The upload service must use client- or server-side input validation to ensure evasion techniques are not used to bypass the whitelist filter.
   * + Only allow alphanumeric file names are allowed for user input – This name will be used as file descriptor / user name
     + Only allow a single extension (Appending a second file type to the file name (e.g. image.jpg.php) would allow attacker to bypass file type restriction)
4. The upload service must set a maximum length for the file name, a maximum size for the file itself, and a maximum user “directory” size to reduce the risk of denial of service attack.
   * An attacker may upload huge files or multiple files to the file space of the Web server which could be exhausted and bring down the website.

### FILE VALIDATION

1. The upload service must perform content checking on all files which are uploaded in a temporary location before being moved to application storage and made available to other users and services.

* The design to achieve this control is to Upload customer’s file new file share; TFS\_AVScan. This new temporary file share (TFS\_AVScan) is a sandbox to be used by all product lines.
  + File Share Configuration: Restricted to only the applications that require access (IFP, BO, SMB and Medicare). And ensure that any files that are awaiting scan may not execute. (execute bit is OFF). Site Ops will define new TFS\_AVSCAN and manage.
  + If the malware scan finds the file is infected then the upload service must not save uploaded files to permanent directory.
    - The technical team decided to move infected files to new file share QFS\_AVSCAN. This new quarantine file share is used by all product lines to quarantine infected files.
    - Log the error as described below. Ensure that file name is logged.
    - Provide a customer error message with options: scan again and reload, fax, call service center.
    - Provide event message to agent via BO. Also provide directions to agent about this situation. BO should also be updated to use this enhanced scan process to ensure that infected files do not bypass scanning and are inadvertently loaded into primary Archive. h
    - The files in QFS\_AVSCAN are not immediately deleted, in case there are issues with the scan engine returning false positives. We can then forward samples to the malware scan vendor.
    - Long term, a platform owned webconnect job may be used to clean up quarantined files after a period of time if not needed for follow-up.
    - The configuration of this file share should match the TFS\_AVSCAN. Restricted to only the applications that require. And ensure that any files that quarantined may not execute. (execute bit is OFF)
    - Site Ops will set-up and manage the new QFS\_AVSCAN.

1. The upload service should scan the file looking for any executable code since the types of documents allowed should have no code. In addition, all uploaded files must be scanned for malware using a robust 3rd party tool with up to date signatures.
   * If scan service is unavailable – reject upload file. Send message to user “Try again later”.
   * The uploaded file could contain malicious code in the form of an exploit, virus, Trojan or malware, which could be used to gain control of the Web server. For example, it is possible to hide PHP code inside an image file and still have it appear to be an image. When the image is opened, it also executes the code hidden in the file. The file could contain scripts or tags that exploit other well-known Web application vulnerabilities, such as cross-site scripting (XSS).
2. Log upload activities for each file uploaded. Data to include in log record
   * date / time,
   * unique identifier (file name) to help with debugging, but avoids HIPAA issues
   * errors encountered – scan failed, malware found, etc.
   * Success / fail
   * file name provided by user, generated file name

### SAVE UPLOADED FILES

1. The directory where files are uploaded must be outside of the website root. Uploaded directory should not have any "execute" permission and all the script handlers must be removed from these directories. If using a filesystem to store uploaded files, an isolated server with a different domain should be deployed.
   * The web server can be compromised by uploading and executing a web-shell which can run commands, browse system files, browse local resources, attack other servers, or exploit the local vulnerabilities.
2. The uploaded file must be renamed according to eHealth convention with visible random name or hidden name prior to completing the upload.
   * If visible name to user then ensure that system filename is random. Recommendation OWASP: Use an algorithm to determine the filenames. For instance, a filename can be a salted MD5 hash of the name of file plus the date of the day. If this method is used, then must consider the case where 2 files generate the same hash to prevent overwriting an existing file.
     + The application should not use the file name supplied by the user or easily guessed name because the uploaded file can then be accessed by entering a specific URL path. This is especially dangerous because the file could be executed immediately after uploading by the attacker.
   * A file could overwrite another file that already exists with the exact same name on the server. If this were a critical file, the new file could cause the website to function incorrectly, or not at all. The new file could be used to deface the website by replacing an existing page, or it could be used to edit the list of allowed file types in order to make further attacks simpler.
3. The files must be encrypted using strong encryption.
   * Minimum 256 bit AES encryption or an equivalent strength protocol for symmetric encryption
   * Minimum 2048 bit RSA or equivalent strength encryption for asymmetric encryption
4. Ensure that uploaded files cannot be accessed by unauthorized users.
   * In 2017, 2 Factor Authentication will be required by users who may view other people’s data.
   * Identify files by carrier to control authorization.

### OPTHER HARDENING

1. Encrypt all transmission of files.
2. Use Cross Site Request Forgery protection methods.
3. Use POST method instead of PUT (or GET!)

Other references:

<https://www.trustwave.com/Resources/SpiderLabs-Blog/Hiding-Webshell-Backdoor-Code-in-Image-Files/>

<https://paragonie.com/blog/2015/10/how-securely-allow-users-upload-files>

<http://www.acunetix.com/websitesecurity/upload-forms-threat/>