Capítulo 2

Python: Archivos - Seguridad

Archivos

- Método POST:
  - Permite cargar archivos de texto y binarios
- Python puede recibir cargas de archivos de cualquier navegador compatible con RFC 1867
  - “Form-based File Upload in HTML”
- Se debe asegurar que el formulario tenga el atributo:

```
enctype="multipart/form-data"
```

Archivos

- Formulario para carga

```
<!DOCTYPE html>
<html>
<body>
   <h1>Ejemplo de upload de archivo</h1>
   <form enctype = "multipart/form-data" action = "archivo.py" method = "post">
      <p>Archivo: <input type = "file" name = "filename" /></p>
      <p><input type = "submit" value = "Enviar archivo" /></p>
   </form>
</body>
</html>
```

Archivos

- Al recibir los datos en el lado del servidor:
  - Recuperar datos desde CGI

```
form = cgi.FieldStorage()
mensaje = ""
if len(form) > 0:
    # obtener archivo
    fileitem = form['filename']
    # Revisar si corresponde a un archivo
    if fileitem.filename:
        # obtenemos el nombre base del archivo sin considerar la ruta completa
        # en el computador del cliente
        fn = os.path.basename(fileitem.filename)
        open('/tmp/' + fn, 'wb').write(fileitem.file.read())
        mensaje = "El archivo " + fn + " fue recibido exitosamente"
    else:
        mensaje = "No se recibió el archivo"
```

Archivos

- Se debe tener cuidado con:
  - Tamaño del archivo
    - Comprobar si hay restricciones de tamaño en configuración de servidor web
    - Verificar tamaños máximos o mínimos en la aplicación
  - Tipo de archivo
    - Aplicaciones permiten solo cierto tipo de archivos
      - Imágenes, Documentos, Archivos comprimidos, etc.
    - No confiar en tipo de archivo informado por cliente
    - Nunca confiar en extensión de nombre de archivo

Archivos

- Revisar tamaño y obtener tipo

```
if fileitem.filename:
    try:
        # obtener tamaño en bytes
        size = os.fstat(fileitem.file.fileno()).st_size
        # averiguar tipo real
        tipo_real = filetype.guess(fileitem.file)
        if size <= MAX_FILE_SIZE:
            # obtenemos el nombre base del archivo sin considerar la
            # ruta completa en el computador del cliente
            fn = os.path.basename(fileitem.filename)
            open('/tmp/' + fn, 'wb').write(fileitem.file.read())
            mensaje = "El archivo {0} fue recibido exitosamente, tamaño {1} tipo {2}"\
                      .format(fn, size, tipo_real.mime)
        else:
            mensaje = "Tamaño de archivo [{0}] excede el máximo [{1}]"\
                      .format(size, MAX_FILE_SIZE)
    except IOError as e:
        mensaje = "error al obtener informacion de archivo {0}: {1}"\
                  .format(e.args[0], e.args[1])
```

Archivos

- Se debe tener cuidado con:
  - Nombre de archivo:
    - No confiar en nombre enviado por cliente:
      - Puede contener caracteres no soportados
      - Puede intentar sobre-escribir archivos existentes
      - Puede ser de largo no soportado
    - Recomendación: Crear un nombre de forma segura en el servidor:
      - Usar funciones de HASH para generar nombre aleatorio y con baja probabilidad de colisión
      - Nombre enviado por cliente se puede almacenar como texto en una base de datos

Archivos

- Recomendaciones:
  - Habilitar carga de archivos en servidor web solo si nuestra aplicación lo requiere
  - Definir:
    - Cantidad máxima de archivos que podemos recibir en una petición HTTP
    - Tamaño máximo de cada archivo en petición HTTP y tamaño de toda la llamada HTTP
    - Tiempo máximo de procesamiento de CGI que recibe archivos
    - Memoria máxima que puede usar CGI que recibe archivos
  - Monitorear continuamente espacio disponibles y estadísticas de sistema de archivos

Archivos

- Recomendaciones
  - Cuidado al borrar archivos que incluyen nombre enviado por cliente
    - ¡Debe validar entrada de datos!
  - Incluya “condiciones del servicio” que el cliente **debe** aceptar para enviar archivos:
    - Considerar archivos con derecho de autor (copyright)
    - Contenido legal dependiendo de territorialidad de la ley aplicable
    - Archivos binarios con fines ilícitos (virus, malware, troyanos, spyware, adware, etc)

Seguridad

- Consejos
  - Nunca se conecte como super usuario o como el propietario de la base de datos
    - Utilice usuarios personalizados con privilegios muy limitados
    - ¿Es necesario tener permisos de escritura en todas las tablas de la base de datos?
    - ¿Se eliminarán datos de esta aplicación?
      - ¿eliminación física o lógica?
  - Use sentencias preparadas con variables asociadas

Seguridad

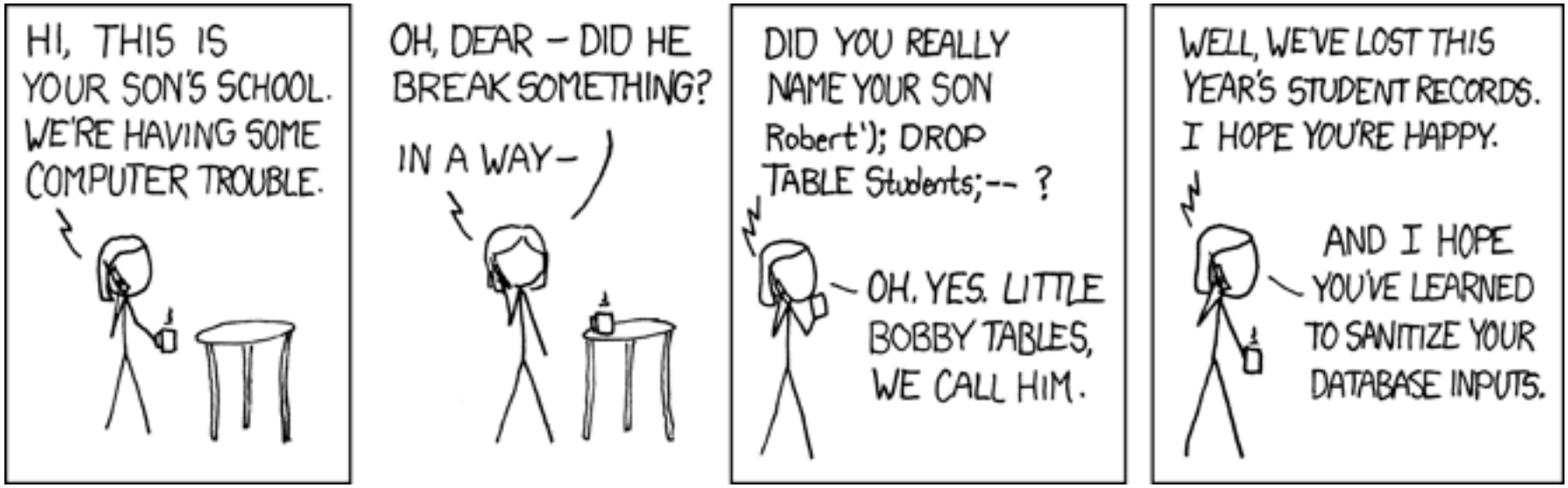

Imagen obtenida desde: http://xkcd.com/327

Seguridad

- Consejos
  - Revise si la entrada proporcionada tiene el tipo de datos que se espera
    - Python provee varias funciones de validación de tipo de datos
  - Si se espera una entrada numérica:
    - verificar los datos con la función isinstance()

```
marks = 90
result = isinstance(marks, int)
if result :
  print("Yes! given variable is an instance of type int")
else:
  print("No! given variable is not an instance of type int")
```