**Dakota State University**

**Pen Tester**

**TextFileGuys**

**Ethan Shafer, Alex Wollman, Chris Johnson**

Table of Contents

[Summary of Project 2](#_Toc351473032)

[Vision Statement 3](#_Toc351473033)

[Functional Requirements 4](#_Toc351473034)

[Non-functional Requirements: 6](#_Toc351473035)

[Design 7](#_Toc351473036)

[Use Cases 8](#_Toc351473037)

[Misuse Cases 16](#_Toc351473038)

# Summary of Project

The goal of the project is to take individual scans from programs such as NMAP as input. Once uploaded the scans are checked for correctness using a file called a Document Type Definition. This DTD is uploaded by users through a plugin structure, described next. Once verified, the scans are combined into one document. This document is formatted in such a way so as to meet the needs of companies desiring security reports. The reports are organized by scans; beginning with the first scan checked and ending with the last.

The program will support NMAP scans by default with a built in DTD. Users can add functionality to the program by uploading their scan's DTDs via the programs plugin structure. This structure allows the user to customize the program, and only concern themselves with scan types they work with. The user creates their DTD, which defines the rules their scan abides by, then uploads it to the program to add support to that scan type. Other types of scans that a user might wish to add to this program include: Nesses, Zap, SET.

To upload scans and DTDs an account must be created with the program. A user can upload scans (which are automatically parsed and checked for correctness,) delete old scans, and create reports. Reports are generated in human readable form and can be viewed or saved on the users machine.

# Vision Statement

The TextFileGuys' project for Secure Software Engineering has various purposes. As a class project, it serves as an assignment and is an effective way to provide experience for the group in software design.

However, at the group level, the project serves as an expansion of knowledge. To us, this project acts as a method to help us learn new technologies while applying them to a real-world project. The PenTester project also serves a dual role as teaching us about security tools and the importance of penetration testing.

# Functional Requirements

1.)Process multiple input data into one output data

1.1)Accept output from tools as input into custom form submission

1.1.1)Input will be accepted from

1.1.1.1)Nmap

1.1.1.2)Nessus

1.1.1.3)ZAP

1.1.1.4)Custom formats

1.1.2)Ensure input is well formed and properly formatted

1.1.2.1)Tags are closed

1.1.2.2)Checked against the plugin DTD

1.1.3)Data not formed correctly will be reported as malformed and rejected

1.2)Store input in database

1.2.1)Break input reports into categories

1.2.1.1)IP

1.2.1.2)Vulnerability

1.2.1.3)Other custom fields

1.3)Correlate data into one form

1.3.1)Pull data from database

1.3.1.1)Pull data in the order that it will appear in the form.

1.3.2)Format data into a chosen file format

1.3.2.2)XML

1.3.2.3)HTML

1.3.2.4)custom

1.3.3)Display Data to screen

1.3.3.1)HTML

2.)Utilize a plugin structure to support additional file formats.

2.1)Create a parser to verify supported and custom Document Type Definition (DTD)

2.1.1)Nmap

2.1.1.1)DTD already exists

2.1.1.1.1)http://nmap.org/book/nmap-dtd.html

2.1.2)Nessus

2.1.2.1)DTD discontinued

2.1.3)Zap

2.1.3.1)Potential DTD

2.1.3.1.1)http://code.google.com/p/zaproxy/source/browse/trunk/src/xml/drivers.dtd

2.1.4)Custom

2.1.4.1)Data organized correctly

2.1.4.2)Attributes listed

2.1.4.3)Entities defined

2.2)Specify directory to upload to DTD

2.2.1)Directory not displayed to user

2.2.2)Stores all DTDs used by program

2.3)Parse custom DTD using general DTD to ensure correctness

2.3.1)Scan type included

2.3.2)Data is present

2.3.3)Data is organized according to DTD

2.4)Use custom DTD to parse scans

2.5)Option to delete DTD from server

2.5.1)Remove file from server

2.5.2)Remove record from database

3.)Account creation and Log-in

3.1)Form for entering data.

3.1.1)Name/Email/etc

3.1.2)Is manager?

3.1.3)Manager email address

3.2)Verify Users

3.2.1)Send Verification code to registered email

3.2.2)Receive verification code to activate account.

3.2.3)Send email to potential manager

3.3)Log-in to the application

4.)Manage user names and passwords

4.1)Change password

4.1.1)Send link to registered email to continue password changing process

4.1.2)Manager ability in program

4.2)Recover password

4.2.1)Send link to registered email to continue password recovery process

4.1.2)Manager ability in program

4.3)Change/Drop respective manager

4.3.1)Manager ability, must verify manager status

4.3.2)Confirm/Drop managed user

5) Ability to search data

5.1) Search data with Keywords used

5.1.1) Search by Metadata

5.1.2) Search Entire Sheet

5.1.3) Multi-search

5.1.3.1) Search by String of words

5.3) Search data by Tools used

5.3.1) Nmap

5.3.2) ZAP

5.3.3) Nessus

5.3.4) Custom

5.4) Search data by User submitted

5.5) ? Search data by Workgroup

# Non-functional Requirements:

1.)Securely store data from external viewing

1.2.1)Safely escape special characters

1.2.2)SQL

2.)Works on the big browsers (chrome, firefox, IE 9+)

3.)secure session handling(login, page traversal, submission, etc.)

4.)Store data (input and output) in a database for future use

# Design

1.)

1.1)

1.1.1)Allow input into a page

1.1.2)Use Regex/string manipulation to convert to a data type

1.1.3)Alert the user to success/failure using response page

1.2)MSSQL database, easy to use with Entity Framework

1.2.3)Chop input using regex/string manipulation

discard extraneous information

1.3)

1.3.1)Entity Framework relates directly to Model classes

1.3.3)Create an HTML template for the form data.

2.)Plugin directory

2.1)load all configuration files in directory

2.1.1)Nessus

2.1.2)Nmap

2.1.3)ZAP

2.1.4)custom

2.2)once loaded, choose config file that matches input file type

2.2.1)if no file type is found to match, reject file

2.3)use config file to verify the input file is valid

2.3.1)tags are ended properly closed

3.)MVC 4/ASP has some built-in user management

3.1)Create web page for input

3.1.2)If manager, give them a 'manager code'

3.1.3)To register under a manager, use 'manager code'

3.1.3.1)Manager must accept user as peon

3.2)Use query parameters to pass verification code/email so user doesn't have to enter it.

Lock account unless verification code is received.

4.)

4.1)Prompt for old password, new password

4.2)Send password reset email with link to reset

4.3)Provide page to edit manager code

4.3.1)If edited and then reverted, manager must reapprove

4.3.2)If edited, new manager must approve

4.4)Provide page with list of pending/current peons

5.)Textboxes with radial buttons/listboxes

Strong use of database here

# Use Cases

Test Case ID: 1.1

Description: Selection of a file and uploading it to a database

Module/Class/Function: /FileUploader/Upload

Test Activities

Step 0(Pre): Assume user is logged in, and valid

Populate Known file formats selection

Dropdown of accepted file types populated

Step 1: Select type of Tool used from dropdown

Step 2: Browse for file (File browser)

Step 3: Select file intended for upload

File path shown outside of file browser

Step 4: Click upload button

File passed to validation

Step 5: Entry created in database

===================================================================

Test Case ID: 1.1.2

Description: Validates the file

Module/Class/Function: FileValidation -- NMAP

Test Activities

Step 0(Pre): Assume file exists

Step 1: Get file passed from uploader

Step 2: Verify NMAP Against DTD

User is prompted of success or failure

Step 3: Return to FileUPload

===================================================================

Test Case ID: 2.0

Description: UploadDTD

Module/Class/Function: Plugin/Upload

Test Activities

Step: 0 (pre)

User is logged in as admin and at management panel

Step: 1

Step Description: Enter PLugin name in "Plugin Name" Field

Expected Results: Name entered in field

Step: 2

Step Description: Select Plugin file from file selector

Expected Results: File selected from file system

Step: 3

Step Description: Sumbit Form

Expected Results: Validate paths and name entered into plugin database

Step: 4

Step Description: Save PLugin file in directory

Expected Results: File exists in server directory

===================================================================

Test Case ID: 2.1.1

Description: CheckNmapDTDCorrectness

Module/Class/Function: Plugin/NmapVerify

Test Activities

Step: 0 (pre)

Step Description: NmapDTD has already been uploaded and a generalDTD exists

Expected Results: None

Step: 1

Step Description: Verify a type field exists

Expected Results: Nmap is present in the first line

Step: 2

Step Description: Verify data fields exist

Expected Results: NmapDTD is not an empty document

Step: 3

Step Description: Save NmapDTD

Expected Results: NmapDTD saved to server directory

===================================================================

Test Case ID: 2.1.2

Description: CheckNessusDTDCorrectness

Module/Class/Function: Plugin/NessusVerify

Test Activities

Step: 0 (pre)

Step Description: NessusDTD has already been uploaded and a generalDTD exists

Expected Results: None

Step: 1

Step Description: Verify a type field exists

Expected Results: Nessus is present in the first line

Step: 2

Step Description: Verify data fields exist

Expected Results: NessusDTD is not an empty document

Step: 3

Step Description: Save NessusDTD

Expected Results: NessusDTD saved to server directory

===================================================================

Test Case ID: 2.1.3

Description: CheckZapDTDCorrectness

Module/Class/Function: Plugin/ZapVerify

Test Activities

Step: 0 (pre)

Step Description: ZapDTD has already been uploaded and a generalDTD exists

Expected Results: None

Step: 1

Step Description: Verify a type field exists

Expected Results: Zap is present in the first line

Step: 2

Step Description: Verify data fields exist

Expected Results: ZapDTD is not an empty document

Step: 3

Step Description: Save ZapDTD

Expected Results: ZapDTD saved to server directory

===================================================================

Test Case ID: 2.1.4

Description: CheckCustomDTDCorrectness

Module/Class/Function: Plugin/CustomVerify

Test Activities

Step: 0 (pre)

Step Description: CustomDTD has already been uploaded and a generalDTD exists

Expected Results: None

Step: 1

Step Description: Verify a type field exists

Expected Results: User supplied name is present in the first line

Step: 2

Step Description: Verify data fields exist

Expected Results: CustomDTD is not an empty document

Step: 3

Step Description: Save CustomDTD

Expected Results: CustomDTD saved to server directory

===================================================================

Test Case ID: 3.1.a

Description: Registration Form as a Manager

Module/Class/Function: /Account/Register

Test Activities

Step: 1

Step Description: Enter Email address in form.

Expected Results: None

Step: 2

Step Description: Enter password and password confirmation.

Expected Results: Password confirmation is validated against the first password.

Step: 3

Step Description: Specify that user is a manger.

Expected Results: None

Step: 4

Step Description: Select Submit.

Expected Results: User receives Verification Email, User Account created in database.

===================================================================

Test Case ID: 3.1.b

Description: Registration Form as an Employee

Module/Class/Function: /Account/Register

Test Activities

Step: 0 (pre)

Step Description: User is on Registration page.

Expected Results: None

Step: 1

Step Description: Enter Email address in form.

Expected Results: None

Step: 2

Step Description: Enter password and password confirmation.

Expected Results: Password confirmation is validated against the first password.

Step: 3

Step Description: Specify that user is \*not\* a manager.

Expected Results: Manager Email prompt shows.

Step: 4

Step Description: Enter Manager Email address.

Expected Results: None

Step: 5

Step Description: Select Submit.

Expected Results: User receives Verification Email, User Account created in database, ManageeApproval is created in database.

===================================================================

Test Case ID: 3.2.a

Description: Verify User Account as a Manager

Module/Class/Function: /Account/ConfirmToken

Test Activities

Step: 0 (pre)

Step Description: User creates Account (Test Case 3.1.a)

Expected Results: User Receives Verification Email to Account Email Address

Step: 1

Step Description: Click link in email

Expected Results: User is taken to Confirmation page, User Account is confirmed in database.

===================================================================

Test Case ID: 3.2.b

Description: Verify User Account as an Employee

Module/Class/Function: /Account/ConfirmToken

Test Activities

Step: 0 (pre)

Step Description: User creates Account (Test Case 3.1.b)

Expected Results: User Receives Verification Email to Account Email Address

Step: 1

Step Description: Click link in email

Expected Results: User is taken to Confirmation page.

User Account is confirmed in database.

Manager receives an email for Confirmation as Employee.

===================================================================

Test Case ID: 4.1

Description: Change User password

Module/Class/Function: /Account/Password

Test Activities

Step: 0 (pre)

Step Description: User logs into Account (Test Case //TODO) and is on /Account page.

Expected Results: User has an active session.

Step: 1

Step Description: User enters Old Password into Form

Expected Results: None

Step: 2

Step Description: User enters New Password and Password Confirmation.

Expected Results: Password Confirmation is validated against the first Password.

Step: 3

Step Description: User clicks Submit.

Expected Results: Successful password change is submitted.

Entry is updated in database.

===================================================================

Test Case ID: 4.3

Description: Change User Manager

Module/Class/Function: /Account/Manager

Test Activities

Step: 0 (pre)

Step Description: User has verified and is logged into an Employee account (Test Cases 3.2.b, //TODO)

User is on /Account page.

Expected Results: None

Step: 1

Step Description: User enters new email address into form.

Expected Results: None

Step: 2

Step Description: User clicks Submit.

Expected Results: User receives confirmation message.

Current Manager relation is deleted from database.

User is dropped from Current Manager's Employee list.

New Manager receives an Email.

===================================================================

Test Case ID: 5.3

Description: Search files by tool(s) used

Module/Class/Function: /File/Search

Test Activities

Step 0(Pre): User is logged in, and validated

User on search page

Step 1: Populate types of tools used

Dropdown of available tools populated

Step 2: Select tool to search by

Step 3: Click Search

Returns file entries that used selected tool

# Misuse Cases

Test Case ID: 1.1.1

Description: Upload of a non-known file type

Module/Class/Function: /FileUploader/Upload

Test Activities

Step 1:

Step Description: User enters unknown file format in upload

Expected Results: File inserted into file browser

Step 2:

Step Description: User Selects file format from known format dropdown

Expected Results: Format box is populated

Step 3:

Step Description: User submits invalid report

Expected Results: Report is not submitted, error message is shown that format does not

match selected.

===================================================================

Test Case ID: 1.4

Description: Upload file larger than 30MB

Module/Class/Function: /FileUploader/Upload

Test Activities

Step 1:

Step Description: User enters large file in upload

Expected Results: File inserted into file browser

Step 2:

Step Description: User submits to large of report

Expected Results: Report is not submitted, error message is shown that file is too large.

===================================================================

Test Case ID: 2.0a

Description: User enters an invalid file name

Module/Class/Function: /Plugin/Upload

Test Activities

Step 0: Assume the user is logged in and at the Plugin Management Page

Step 1:

Step Description: User enters a file name containing illegal special characters (‘)

Expected Results: None

Results: None

Step 2:

Step Description: User selects a file to upload

Expected Results: File Browser window pops up and file is selected

Results: File Browser window pops up and file is selected

Step 3:

Step Description: User presses submit.

Expected Results: The file name is properly escaped to avoid database problems

===================================================================

Test Case ID: 2.0b

Description: User uploads an inappropriate DTD file

Module/Class/Function: /Plugin/Upload

Test Activities

Step 0: Assume the user is logged in and at the Plugin Management Page

Step 1:

Step Description: User enters a proper file name

Expected Results: None

Results: None

Step 2:

Step Description: User selects a file to upload that is larger than 30 Mbps.

Expected Results: File Browser window pops up and file is selected

Results: File Browser window pops up and file is selected

Step 3:

Step Description: User presses submit.

Expected Results: The file is rejected and an error is returned to the user.

===================================================================

Test Case ID: 2.1.1

Description: Improper Nmap DTD uploaded

Module/Class/Function: /Plugin/NmapVerify

Test Activities

Step 0: NmapDTD has already been uploaded and a generalDTD exists

Step 1:

Step Description: NmapDTD type field is not present

Expected Results: generalDTD flags an error and bubbles it up

Step 2:

Step Description: Display error to user

Expected Results: Display error to user

Results

===================================================================

Test Case ID: 2.1.2

Description: Improper Nessus DTD uploaded

Module/Class/Function: /Plugin/NessusVerify

Test Activities

Step 0: NessusDTD has already been uploaded and a generalDTD exists

Step 1:

Step Description: NessusDTD type field is not present

Expected Results: generalDTD flags an error and bubbles it up

Step 2:

Step Description: Display error to user

Expected Results: Display error to user

===================================================================

Test Case ID: 2.1.3

Description: Improper Nmap DTD uploaded

Module/Class/Function: /Plugin/ZapVerify

Test Activities

Step 0: ZapDTD has already been uploaded and a generalDTD exists

Step 1:

Step Description: ZapDTD type field is not present

Expected Results: generalDTD flags an error and bubbles it up

Step 2:

Step Description: Display error to user

Expected Results: Display error to user

Results

===================================================================

Test Case ID: 2.1.4

Description: Improper Custom DTD uploaded

Module/Class/Function: /Plugin/CustomVerify

Test Activities

Step 0: CustomDTD has already been uploaded and a generalDTD exists

Step 1:

Step Description: CustomDTD type field is not present

Expected Results: generalDTD flags an error and bubbles it up

Step 2:

Step Description: Display error to user

Expected Results: Display error to user

Results

===================================================================

Test Case ID: 3.1.a

Description: User enters an invalid email Address

Module/Class/Function: /Account/Register

Test Activities

Step 1:

Step Description: User enters an invalid email address in form

Expected Results: none

Step 2:

Step Description: User fills out rest of form correctly and clicks submit.

Expected Results: Form does not submit and email address is flagged as invalid.

User has to re-enter password but no other information.

===================================================================

Test Case ID: 3.1.b

Description: User enters an invalid password (not long/complex enough)

Module/Class/Function: /Account/Register

Test Activities

Step 1:

Step Description: User enters an invalid password in form

Expected Results: none

Step 2:

Step Description: User fills out rest of form correctly and clicks submit.

Expected Results: Form does not submit and password is flagged as weak.

User has to re-enter password but no other information.

===================================================================

Test Case ID: 3.1.c

Description: User enters an incorrect password confirmation

Module/Class/Function: /Account/Register

Test Activities

Step 1:

Step Description: User enters password in first ‘Password’ box.

Expected Results: none

Step 2:

Step Description: User enters different password in ‘Confirm Password’ box.

Expected Results: none

Step 3:

Step Description: User fills out rest of form correctly and clicks submit.

Expected Results: Form does not submit and password is flagged as incorrect.

User has to re-enter password but no other information.

===================================================================

Test Case ID: 3.1.d

Description: User does not fill out registration form completely

Description: /Account/Register

Test Activities

Step 1:

Step Description: User does not fill out a portion of the form.

Expected Results: none

Step 2:

Step Description: User fills out rest of form correctly and clicks submit.

Expected Results: Form does not submit and password is flagged as weak.

User has to re-enter password and any other missed information.

===================================================================

Test Case ID: 3.1.e

Description: User enters an invalid password (not long/complex enough)

Description: /Account/Register

Test Activities

Step 1:

Step Description: User enters an email address for a manager that is not in our system.

Expected Results: none

Step 2:

Step Description: User fills out rest of form correctly and clicks submit.

Expected Results: Form submits, user is created, email is not sent to the entered email address.

===================================================================

Test Case ID: 3.2

Description: User confirms account

Description: /Account/ConfirmToken

Test Activities

Step 1:

Step Description: User changes the URL from the confirmation email.

Expected Results: Page prompts with incorrect verification information.

===================================================================

Test Case ID: 3.2

Description: User confirms account

Description: /Account/Login

Test Activities

Step 1:

Step Description: Enters incorrect login information, clicks submit.

Expected Results: Returned to login page with generic failure message.

Step 2:

===================================================================

Test Case ID: 4.1

Description: User confirms account

Description: /Account/Password

Test Activities

Step 1:

Step Description: User changes the URL from the confirmation email.

Expected Results: Page prompts with incorrect verification information.

===================================================================

Test Case ID: 4.3

Description: User changes manager to invalid manager email address.

Description: /Account/Manager

Test Activities

Step 1:

Step Description: User enters an email address for a manager that is not in our system.

Expected Results: none

Step 2:

Step Description: User clicks submit.

Expected Results: Email is not sent to the incorrect email address.

===================================================================

Test Case ID: 5

Description: SQL injections

Description: /File/Search

Test Activities

Step 1:

Step Description: Attempt to inject SQL statement in the input box.

Expected Results: SQL Injection is aborted. User sees invalid search criteria message.