**Dakota State University**

**Pen Tester**

**TextFileGuys**

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

Contents

[Summary of Project 1](#_Toc351454885)

[Vision Statement 3](#_Toc351454886)

[Functional Requirements 4](#_Toc351454887)

[Non-functional Requirements: 6](#_Toc351454888)

[Design 7](#_Toc351454889)

# 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