User Stories for CSCD 350 Spring 2024

ScanMasterX

V 1.0

Team #10: UwUltimate Stardust Crusaders

Submitted By

Lewis Thomas	Lthomas32@ewu.edu
William Reese	Wreese1@ewu.edu
Eric Leachman	Eleachman04@ewu.edu
Dennis Vinnikov	dvinnikov@ewu.edu
Alexa Darrington	adarrington@ewu.edu

Instructor: Sanmeet Kaur

GSA: Harley Davis, Dominic MacIsaac

Lab Section: Rm 219 – Section 2

Date: 04/19/2024

GitHub Repository: https://github.com/Sanmeet-EWU/github-teams-project-bid-uwultimate-stardust-crusaders/tree/main

Section 1 – User Stories

U1 – GUI to interact with the program.

As a customer, I would like a user interface to interact with the program easily and obscure details.

Assumptions and Details:

- We will have a GUI with menus and comprehensive prompts and output.
- We will use Tkinter and subnetting to achieve this.

Acceptance Criteria:

Given a GUI with a menu of options

And a user deciding what to test

When they choose an option

Then our GUI program will display desired information.

U2 – Add a login system.

As a user of this tool, I would like a login system to keep track of my information.

Assumptions and Details:

- We possibly will have a database to store credentials.
- We maintain and store credentials and password hashes.

Acceptance Criteria:

Given 100 users in our system

And they all login to our GUI

When we look in our database

Then we should have 100 user logins.

U3 – Component to display network topology.

As a customer, I would like a component to display my network's topology.

Assumptions and Details:

- We will scan using NMAP and visualize and display the output for the user.
- This will be a foundational component within our program.

Acceptance Criteria:

Given some network information

And the NMAP framework

When our tool maps out the topology

Then it will be displayed in a visualized format.

U4 - Component to scan multiple subnets.

As a business owner, I want to be able to scan multiple subnets with this tool to cover all my bases.

Assumptions and Details:

- We will gather information from the user to scan all possible subnets.
- NIST, MITRE and Metasploit will be used.

Acceptance Criteria:

Given a businesses' subnets

And our programs use of frameworks

When they want to see vulnerable ports

Then our program will provided that output

U5 – Component to view server port vulnerabilities.

As a Business owner, I want to be able to visually see any vulnerabilities my server ports might have, so that I can identify any security issues I might need to fix.

Assumptions and Details:

- The software is readily available to new servers.
- The services are services that already have security details available.

Acceptance Criteria:

Given the program

When I set up a server

Then the program should be able to enumerate over every port.

U6 – Ability to scan a client for protocol security risks.

As a home network enthusiast, I want to ensure I am using secure protocols on my systems, so that I can have peace of mind that my home network is secure.

Assumptions and Details:

- The protocols are commonly known protocols that have been tested.

Acceptance Criteria:

Given this program, When I scan my systems, Then the scan should give information on the protocols I am running on my systems.

U7 – Visualization of network vulnerabilities.

As a grandma who knows nothing about the internet, I want a visual product that can give me feedback on how vulnerable my home network is, so that I can be more educated and cautious about my internet habits.

Assumptions and Details:

- The software is pre-installed.
- The software is intuitive enough for the technologically non-savvy to use.

Acceptance Criteria:

Given this program

When I run the program

Then some information is construed in a clear decisive manner so that lay persons can make decisions and be educated.

U8 – Interfaces for commonly used cybersecurity tools.

As a student who doesn't like using CLI programs, I want an easy-to-use GUI program that allows me to use common cyber security tools, so that I can stop using ChatGPT to write the commands for me.

Assumptions and Details:

- Students like the software better than AI.
- The software is intuitive enough to give up CLI for it.

Acceptance Criteria:

Given this program.

When I need to enumerate several security solutions.

Then the program gives a readily accessible tool set.

U9 – Component to intuitively crack hashes.

As an Information Security Lead, I need to test the organization's passwords so that I can identify Attack Vectors.

Assumptions and Details:

- We have access to everyone’s password hash.
- We are testing against common word list.

Acceptance Criteria:

Given there are 100 employees in Active Directory

And Security Operations have agreed to give us access

When I request passwords

Then I should receive 100 hashes.

U10 – Implement a hash identification module.

As an Information Security Analyst, I need to identify a hash so that I can send it to the password cracking team.

Assumptions and Details:

- The hashes will be commonly identifiable hashes.

Acceptance Criteria:

Given a password hash

And password is hashed with an approved hash

When I receive the password

Then I should output the type.

U11 – Implement a component to run commonly used exploits.

As a developer, I need to identify my vulnerable machines so that I can prevent remote access.

Assumptions and Details:

- I will only be able to exploit known exploits.
- It will either pass or fail.

Acceptance Criteria:

Given a virtual machine

And virtual machine is on the same network

When I test the machine

Then I should either be given access or fail.

U12 – Calculate and display the time it took to crack a hash.

As an IAM manager, I need to display times of password cracking so that I can present the data in an audit.

Assumptions and Details:

- Passwords are already cracked.
- Types are already known.

- Times are included in the list.

Acceptance Criteria:

Given a list of password hashes

And the passwords are cracked and timed

When I upload the list

Then I should receive a graph displaying the data.

U13 – Lockout user from account if password is attempted too many times.

As a potentially vulnerable user, I would like it if there was a system in place that prevented brute force attacks on my Scanmaster X account.

Assumptions and Details:

- Potentially malicious users what to steal our user's credentials.

Acceptance Criteria:

Given an at risk user

And their account is under attack

When the attacker fails multiple logins

Then they will be unable to attempt anymore logins.

U14 – Functionality to create a new account.

As a prospective customer, I would like to create a new account so that I can use the Scanmaster X software.

Assumptions and Details:

- The user is interested in signing up for the Scanmaster X software.

Acceptance Criteria:

Given a prospective customer

And they wish to make an account

When the user accesses the software

Then they will be able to create a new account.

U15 – Functionality to delete an existing account.

As a disgruntled user, I no longer wish to have my Scanmaster X account, so I would like to delete it.

Assumptions and Details:

- The user has lost interest in the Scanmaster X software.
- Lewis might stink?

Acceptance Criteria:

Given a disgruntled user

And they wish to delete their account

When the user goes to their settings

Then they will be able to delete their account.

U16 – Ability to identify vulnerabilities on a server.

As a business owner, I would like to be able to easily scan vulnerabilities on my server.

Assumptions and Details:

- The user would like to get a detailed report on how to better secure their servers.

Acceptance Criteria:

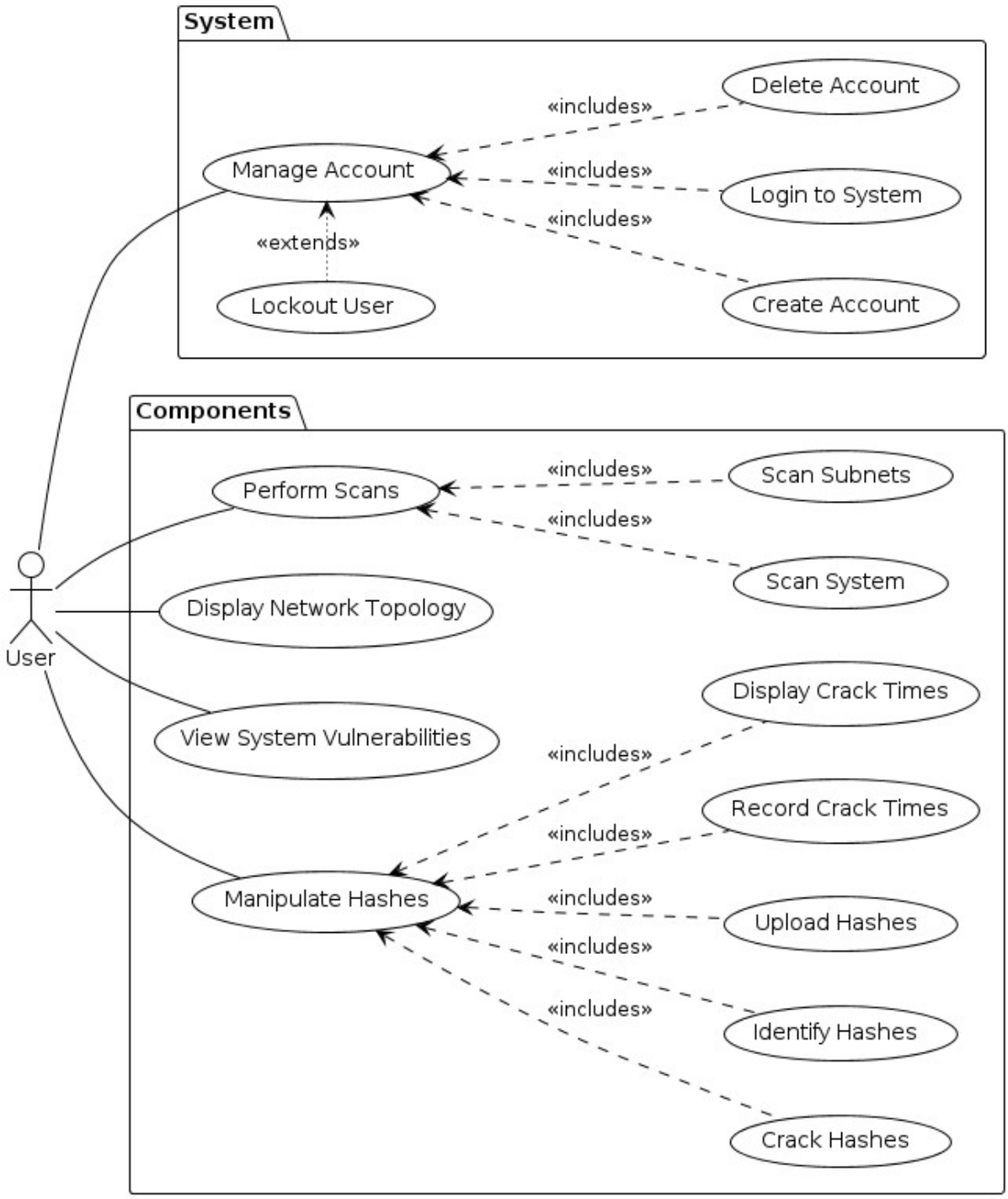
Given a server owner

And they wish to secure their servers

When they run a scan on their server

Then they will get a detailed report of potential vulnerabilities.

Section 2 – Use Case Diagram



Section 3 – Requirements & Specifications

Section 3.1 – Account Management Requirements

[Req 1, U14 – Create Account] The user must be able to make a new account.

- **[Spec 1.1]** The user must provide a valid email address during account creation.
- **[Spec 1.2]** The user's password must be more than 12 characters long, include a mix of capital and lower-case letters, cannot have more than 3 repeating characters in a row, and must include at least 1 special character.

[Req 2, U2 – Login to Account] The user must be able to login to their account.

[Req 3, U15 – Delete Account] The user must be able to delete their account.

[Req 4, - U13 – Login Threshold] After the threshold for failed password attempts has been exceeded, the user's account is to be locked out.

[Req 5, U2 – Login Help] If the user wishes to regain access to their account, they must contact customer support. The customer support contact information will be provided once they are locked out of their account.

[Req 6, U2 – Change Password] The user must be allowed to change their password.

[Req 7, U2 – Reinstate Account] There must be the administrative ability to reinstate user's accounts once they have been locked out.

Section 3.2 – Hash-cracking Component Requirements

[Req 8, U9 – Crack Hash] There must be a component that allows the user to crack hashes.

- **[Spec 8.1]** The user will be allowed to provide a list of hashes.
- **[Spec 8.2]** The user will be allowed to provide a file of hashes.
- **[Spec 8.3]** The user will be allowed to provide a singular hash.

[Req 9, U9 – Wordlist Crack] The hash cracking component must be able to compare password hashes to a wordlist.

- **[Spec 9.1]** Commonly used wordlists are to come packaged with the Scanmaster X software.
- **[Spec 9.2]** Users may choose any of the preinstalled wordlists when they are cracking a hash.

[Req 10, U10 – Identify Hash] The hash cracking component must have an option to identify the hash-type of any given-hash.

- **[Spec 10.1]** If a valid hash-type cannot be extracted from the provided hash, an appropriate error message will be given to the user.

[Req 13, U12 – Time to Crack] The time it took to crack each hash must be viewable by the user.

[Req 14, U12 – Crack Progress] A progress tracker of the current hashes being cracked must be displayed to the user while the hash is being cracked.

[Req 15, U9 – Hash Strength] There must be a component that allows the user to view the strength of a given hash.

- **[Spec 15.1]** The user may choose one of the preinstalled wordlists to validate the strength of their hash.
- **[Spec 15.2]** The user may choose to run all the preinstalled wordlists to validate the strength of their hash.
- **[Spec 15.3]** If the hash fails the validation, then the time it took to crack the hash will be provided.

Section 3.2 – Server Vulnerability Identifier Component

[Req 11, U11, U16 – Vulnerability Identifier] There must be a feature to identify vulnerabilities on a server.

- **[Spec 11.1]** The publicly available Metasploit API will be used to launch exploits and consequentially identify vulnerabilities.
- **[Spec 11.2]** The user may choose to run the vulnerability scanner in terse mode, meaning that the whole test will end once a single vulnerable port is found.

[Req 12, U16 – Vulnerability Report] A sufficiently (see specifications below) detailed report of what vulnerabilities exist on a given device must be provided after running the vulnerability identifier component.

- **[Spec 12.1]** The report will contain the following information:
    - The type of attack which was used.
    - Each variation of parameters that were used in the attack.
    - The result of each variation of the attack that was ran.
    - Potential actions the user can take to secure their device.
    - (Optional) A link to the relevant vulnerability within the NVD.

[Req 16, U11, U16 – Attack Sequences] A minimum of 3 professionally approved and curated (Lewis) attack sequences must be provided as selectable options to the user.

- **[Spec 16.1]** An attack sequence shall be defined as a logical sequence of attacks, whose parameters and general flow is to be determined by the previously ran attack in the sequence, and whose ultimate purpose is to ‘pwn’ the user’s machine.
- **[Spec 16.2]** Lewis shall henceforth be referred to as a cybersecurity professional and touted to the utmost degree to gratify the claims of the extreme prestige of this Scanmaster X component.

Section 3.3 – Network Security Component Requirements

[Req 17, U5 – Port Evaluation] There must be a component that will evaluate the security of the ports on a given server.

- **[Spec 17.1]** The NMAP API will be used to verify whether an open port has the potential to be exploited.

[Req 18, U5, U6 – Port Selection] The user must be able to select which ports, or which group of ports, that they wish to test.

- **[Spec 18.1]** A list of the most used ports will be provided to the user as a selectable option for the scan.

[Req 19, U5, U7 – Port Scan Report] After a port scan, the user must be provided with a report of any vulnerabilities that were found on the given device.

- **[Spec 19.1]** As a part of the report, the user will be provided with prospective step-by-step solutions on how to secure the given port.

[Req 20, U5 – Port Selection 2] The user will be able to specify which device(s) whose and which ports to scan.

- **[Spec 20.1]** The user will have the option to scan their local device.
- **[Spec 20.2]** The user will have the option to scan their local network.
- **[Spec 20.3]** The user will have the option to scan the provided IP address.
- **[Spec 20.4]** The user will have the option to scan a list of IP addresses.
- **[Spec 20.5]** The user will have the option to scan all IP addresses within a provided file.

[Req 21, U4 – Subnet Scanner] The port scanning component must have the option to scan subnets.

- **[Spec 21.1]** The user will have the option to provide multiple subnets.
- **[Spec 21.2]** The user will have the option to provide file of all subnets they wish to scan.

[Req 22, U3 – Network Topology] There must be a component to view the network topology of a given IP address.

- **[Spec 22.1]** The provided IP address will be given in CIDR notation.
- **[Spec 22.2]** The NMAP API will be used to logically scan each host within the target network.

Section 3.4 – General Requirements

[Req 23, U8 – Common Tools] There must be a list of commonly used cyber-security tools provided to the user.

[Req 24, U8 – Common Tools] The commonly used cyber-security tools must be sortable by category.

Section 4 – Glossary

Brute force: a hacking method that uses trial and error to crack passwords, login credentials, and encryption keys.

CIDR notation: a way to represent IP address and a suffix of the network (ex. 192.168.1.1/24)

Credential: a set of login or authentication data that verify a user's identity and grant them access to a particular system or service.

Exploit: a segment of code or a program that maliciously takes advantage of vulnerabilities or security flaws in software or hardware.

GUI: graphical user interface, a way for user to interact with the app by manipulating graphical elements such as icons, buttons, sliders and menus.

Hash: a one-way mathematical function that turns data into a string of nondescript text that cannot be reversed or decoded.

IAM: Identity Access Management

Network Security: the protection of the underlying networking infrastructure from unauthorized access, misuse, or theft.

Network Topology: the physical and logical arrangement of nodes and connections in a network.

NMAP: Network Mapper, a free and open-source tool used for vulnerability checking, port scanning, and network mapping.

PWN: to conquer or dominate, often means that your account or system has been breached, or password have been stolen.

Server: a computer or computer program which manages access to a centralized resource or service in a network.

Subnet: a part of a larger network such as the internet.

Tkinter: a python interface for the Tk GUI toolkit.

Tk GUI toolkit: a cross-platform widget toolkit that provides a library of basic elements of GUI widgets for building a graphical user interface in many programming languages.

Virtual Machine: a computer system created using software on one physical computer in order to emulate the functionality of another separate physical computer.