



***Frozen Network Reporting (FNR):
Senior Design Project Documentation***

Florida International University – Summer 2025

Product Owner/Team Lead: Kevin Ocana

Team Members: Elizabeth Luzardo, Marek Ferfecky, Anthony Torres

Instructor: Masoud Sadjadi

Table of Contents

- Introduction3
 - Project Overview3
 - Purpose3
 - Problem Motivation.....3
 - System Design.....3
 - Functionality and Workflow.....3
 - Setup and Installation4
 - Testing and Evaluation.....4
 - Limitations and Concerns4
 - Future Enhancements.....4
- Risk Matrix5
 - Risk Details5
- Conclusion6
- Sprint Planning:7
 - 1 Sprint Planning Meeting Minutes:7
 - 2 Sprint Planning Meeting Minutes:8
 - 3 Sprint Planning Meeting Minutes:9
 - 4 Sprint Planning Meeting Minutes: 10
 - 5 Sprint Planning Meeting Minutes: 11
- Daily Scrum Meeting: 12
- APPENDIX A..... 15
 - README.txt 15
- APPENDIX B..... 17
 - Code 17

Introduction

Project Overview

Frozen Network Reporting (FNR) is a Python-based vulnerability scanner designed to identify network security weaknesses. It scans open ports, services, and configuration issues using tools like **Nmap** and matches findings against known vulnerabilities in the CVE database. The goal is to proactively identify threats before they are exploited.

Purpose

With cyberattacks growing in volume and complexity, organizations face serious risks from misconfiguration, outdated systems, and exposed services. FNR helps mitigate these offering automated scanning and actionable insights.

Problem Motivation

The project was initiated in response to the increasing number of cyberattacks targeting network infrastructure across both public and private sectors. Common vulnerabilities such as outdated systems, misconfigured services, and unsecured wireless networks are often exploited by malicious actors. Organizations frequently lack visibility into these issues until a breach occurs. FNR was created to proactively identify these risks and assist in remediation before damage can take place.

System Design

FNR is built using a frontend-backend model. The frontend, developed with HTML, CSS, and JavaScript, provides a user-friendly interface that displays scanning results, risk severity, and options for user interaction. The backend is implemented using Python and Flask, which handles API communication and scanning logic. It incorporates Nmap and socket-based network communication to perform automated scans. Streamlit is used to provide a lightweight and interactive interface for initiating scans and exporting results in PDF format. Visualization is handled through Chart.js, which helps users understand the scope and severity of discovered vulnerabilities.

Functionality and Workflow

The scanner begins by identifying hosts on a network using ICMP ping and ARP discovery. Once active hosts are found, the scanner performs comprehensive TCP and UDP scans to identify open

ports and the services running on them. These findings are then analyzed against known vulnerabilities using the CVE database. Users can run scans, review results in a browser, and export findings for further documentation. A test suite ensures port scan accuracy, validates CVE matches, and checks for performance under high loads.

Setup and Installation

FNR includes a batch script, `setup_scanner.bat`, which simplifies installation on Windows systems. It installs required Python dependencies, ensures Nmap is installed and accessible via system PATH, and prepares the scanner for execution. The Python dependencies are listed in the `requirements.txt` file, which includes Flask for web handling, `python-Nmap` for scanner integration, and Requests for API handling.

Testing and Evaluation

The scanner was tested across various categories. Functional testing involved verifying accurate port detection and successful CVE matching. Performance testing included scanning large IP ranges to test stability under load. UI and UX testing were conducted by collecting feedback from administrative users to ensure clarity and ease of use. The scanner's stealth was also tested to evaluate whether it could be detected by intrusion prevention systems during real-time scans.

Limitations and Concerns

While FNR provides accurate scanning and helpful insights, it has some limitations. False positives and negatives may occur, especially with newer threats not yet included in the CVE database. Additionally, changes in network protocols may affect scan effectiveness. For privacy and compliance reasons, the tool only scans public network data and avoids deep inspection of sensitive internal information.

Future Enhancements

The team is considering expanding the scanner's functionality by adding features such as scheduled scans, real-time alerts, and log export capabilities. Publishing the application as an open-source tool or web-accessible app is also under discussion. A future version may include a fully custom-built scanning engine for even greater control and accuracy.

Risk Matrix

Probability ↑	Impact →	Very Low (1)	Low (2)	Medium (3)	High (4)	Very High (5)
	Very High (5)	Team Conflicts	CVE API Issues	False Positives	Performance Issues	Security Bypass
	High (4)	UI/UX Issues	Installation Problems	Protocol Changes	Legal/Compliance	Network Detection
	Medium (3)	Documentation	Export Features	Dependency Issues	Scalability	Data Privacy
	Low (2)	Code Quality	Version Control	Testing Coverage	Maintenance	Nmap Dependency
	Very Low (1)	Hardware Issues	OS Compatibility	Third-party APIs	Team Availability	Zero-day Exploits

Risk Details

- **False Positives/Negatives (Probability: 5, Impact: 3)** Scanner may produce inaccurate results, especially with newer threats not in CVE database
- **Security Bypass by Malicious Users (Probability: 5, Impact: 5)** Tool could be used maliciously for unauthorized network reconnaissance
- **CVE Database API Issues (Probability: 5, Impact: 2)** External CVE API may be unavailable or rate-limited
- **Performance Issues Under Load (Probability: 5, Impact: 4)** Scanner may fail or become unstable when scanning large IP ranges

- **Team Conflicts on Functionality (Probability: 5, Impact: 1)** Disagreements on core app functionality as mentioned in scrum notes
- **Network Detection by IPS (Probability: 4, Impact: 5)** Scanner activity detected by intrusion prevention systems
- **Legal/Compliance Violations (Probability: 4, Impact: 4)** Unauthorized scanning could violate laws or organizational policies
- **Network Protocol Changes (Probability: 4, Impact: 3)** Changes in network protocols may affect scan effectiveness
- **Installation Problems (Probability: 4, Impact: 2)** setup_scanner.bat may fail on different Windows configurations
- **Nmap Dependency Risk (Probability: 2, Impact: 5)** Critical dependency on Nmap availability and proper installation
- **Data Privacy Concerns (Probability: 3, Impact: 5)** Potential exposure of sensitive network information during scanning
- **Scalability Limitations (Probability: 3, Impact: 4)** System may not handle enterprise-level scanning requirements

Conclusion

Frozen Network Reporting is a proactive cybersecurity tool that helps users and organizations detect vulnerabilities in their network environment. With its multi-layered scanning strategy, intuitive interface, and CVE integration, FNR simplifies vulnerability management and improves threat visibility. Built with open-source tools and modern frameworks, the scanner is reliable, accessible, and designed to meet the real-world demands of cybersecurity analysis.

Sprint Planning:

1 Sprint Planning Meeting Minutes:

Attendees: Anthony Torres, Marek Ferfecky, Elizabeth Luzardo, Kevin Ocana

Start time: **8:20pm**

End time: **8:50pm**

After discussion, the velocity of the team was estimated to be 20.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- Research = 4 Story Points
- Project organization/visuals = 12 Story Points
- Debate = 4 Story Points

The team members indicated their willingness to work on the following user stories.

- Anthony Torres
 - Research = 1 Story Points
 - Project organization/visuals = 3 Story Points
 - Debate = 1 Story Points
- Marek Ferfecky
 - Research = 1 Story Points

-
-
- - Project organization/visuals = 3 Story Point
 - Debate = 1 Story Point
- Elizabeth Luzardo
 - Research = 1 Story Points
 - Project organization/visuals = 3 Story Points
 - Debate = 1 Story Point
- Kevin Ocana
 - Research = 1 Story Points
 - Project organization/visuals = 3 Story Points
 - Debate = 1 Story Points

2 Sprint Planning Meeting Minutes:

Attendees: Anthony Torres, Marek Ferfecky, Elizabeth Luzardo, Kevin Ocana

Start time: **6:45pm**

End time: **7:15pm**

After discussion, the velocity of the team was estimated to be 20.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- Research = 12 Story Points
- Project organization/visuals = 4 Story Points
- Debate = 4 Story Points

- Anthony Torres
 - Research = 3 Story Points
 - Project organization/visuals = 1 Story Points
 - Debate = 1 Story Points
- Marek Ferfecky
 - Research = 3 Story Points

- Project organization/visuals = 1 Story Point
- Debate = 1 Story Point

- Elizabeth Luzardo

- Research = 3 Story Points

The team members indicated their willingness to work on the following user stories.

- Project organization/visuals = 1 Story Points
- Debate = 1 Story Point

- Kevin Ocana

- Research = 3 Story Points
- Project organization/visuals = 1 Story Points
- Debate = 1 Story Points

3 Sprint Planning Meeting Minutes:

Attendees: Anthony Torres, Marek Ferfecky, Elizabeth Luzardo, Kevin Ocana

Start time: **5:30 pm**

End time: **6:00 pm**

After discussion, the velocity of the team was estimated to be 20.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- Research = 12 Story Points
- Project organization/visuals = 4 Story Points
- Debate = 4 Story Points

The team members indicated their willingness to work on the following user stories.

- Anthony Torres

- Research = 3 Story Points
- Project organization/visuals = 1 Story Points
- Debate = 1 Story Points

- Marek Ferfecky

- Research = 3 Story Points

- Project organization/visuals = 1 Story Point
- Debate = 1 Story Point
- Elizabeth Luzardo
 - Research = 3 Story Points
 - Project organization/visuals = 1 Story Points
 - Debate = 1 Story Point
- Kevin Ocana
 - Research = 3 Story Points
 - Project organization/visuals = 1 Story Points
 - Debate = 1 Story Points

4 Sprint Planning Meeting Minutes:

Attendees: Anthony Torres, Marek Ferfecky, Elizabeth Luzardo, Kevin Ocana

Start time: **5:30 pm**

End time: **6:00 pm**

After discussion, the velocity of the team was estimated to be 20.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- Research = 12 Story Points
- Project organization/visuals = 4 Story Points
- Debate = 4 Story Points

The team members indicated their willingness to work on the following user stories.

- Project organization/visuals = 1 Story Points
- Anthony Torres
 - Research = 3 Story Points
 - Project organization/visuals = 1 Story Points
 - Debate = 1 Story Points
- Marek Ferfecky
 - Research = 3 Story Points

- Project organization/visuals = 1 Story Point
- Debate = 1 Story Point
- Elizabeth Luzardo
 - Research = 3 Story Points
 - Debate = 1 Story Point
- Kevin Ocana
 - Research = 3 Story Points
 - Project organization/visuals = 1 Story Points
 - Debate = 1 Story Points

5 Sprint Planning Meeting Minutes:

Attendees: Anthony Torres, Marek Ferfecky, Elizabeth Luzardo, Kevin Ocana

Start time: **4:30 pm**

End time: **5:00 pm**

After discussion, the velocity of the team was estimated to be 20.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

- Research = 12 Story Points
- Project organization/visuals = 4 Story Points
- Debate = 4 Story Points

The team members indicated their willingness to work on the following user stories.

- Anthony Torres
 - Research = 3 Story Points
 - Project organization/visuals = 1 Story Points
 - Debate = 1 Story Points
- Marek Ferfecky
 - Research = 3 Story Points

- Project organization/visuals = 1 Story Point
- Debate = 1 Story Point
- Elizabeth Luzardo
 - Research = 3 Story Points
 - Project organization/visuals = 1 Story Points
 - Debate = 1 Story Point
- Kevin Ocana
 - Research = 3 Story Points
 - Project organization/visuals = 1 Story Points
 - Debate = 1 Story Points

Daily Scrum Meeting:

Daily Scrum Meeting Minutes:

Date: 07/25/2025

Attendees: Anthony Torres, Marek Ferfecky, Elizabeth Luzardo, Kevin Ocana

Start time: **8:45 pm**

End time: **11:45 pm**

Anthony Torres:

- How many hours did I work since the last meeting? ○ 3 hours
- What was done since the last scrum meeting?
 - We continued making significant improvements to our website, focusing on user experience enhancements and visual design elements. The team dedicated considerable time to adding final touches, including content refinement, layout adjustments, and ensuring all features function properly.
- What is planned to be done until the next scrum meeting?
 - For our upcoming meeting, we will continue the ongoing process of refining our website by addressing any remaining technical issues and optimizing performance. Additionally, we plan to start working on poster development.
- What are the hurdles?
 - A hurdle our team could face during the meeting is disagreements on the core functionality of the app, as well implementation of those core functionalities.

Marek Ferfecky:

- How many hours did I work since the last meeting? ○ 3 hours
- What was done since the last scrum meeting?
 - We continued making significant improvements to our website, focusing on user experience enhancements and visual design elements. The team dedicated considerable time to adding final touches, including content refinement, layout adjustments, and ensuring all features function properly.
- What is planned to be done until the next scrum meeting?
 - For our upcoming meeting, we will continue the ongoing process of refining our website by addressing any remaining technical issues and optimizing performance. Additionally, we plan to start working on poster development.
- What are the hurdles?
 - A hurdle our team could face during the meeting is disagreements on the core functionality of the app, as well implementation of those core functionalities.

Elizabeth Luzardo:

- How many hours did I work since the last meeting? ○ 3 hours
- What was done since the last scrum meeting?
 - We continued making significant improvements to our website, focusing on user experience enhancements and visual design elements. The team dedicated considerable time to adding final touches, including content refinement, layout adjustments, and ensuring all features function properly.

- What is planned to be done until the next scrum meeting?
 - For our upcoming meeting, we will continue the ongoing process of refining our website by addressing any remaining technical issues and optimizing performance. Additionally, we plan to start working on poster development.
- What are the hurdles?
 - A hurdle our team could face during the meeting is disagreements on the core functionality of the app, as well implementation of those core functionalities.

Kevin Ocana:

- How many hours did I work since the last meeting? ○ 3 hours
- What was done since the last scrum meeting?
 - We continued making significant improvements to our website, focusing on user experience enhancements and visual design elements. The team dedicated considerable time to adding final touches, including content refinement, layout adjustments, and ensuring all features function properly.
- What is planned to be done until the next scrum meeting?
 - For our upcoming meeting, we will continue the ongoing process of refining our website by addressing any remaining technical issues and optimizing performance. Additionally, we plan to start working on poster development.
- What are the hurdles?
 - A hurdle our team could face during the meeting is disagreements on the core functionality of the app, as well implementation of those core functionalities.

APPENDIX A.

README.txt

Networkscanner - How To Use

=====

This guide explains how to set up and use the `Networkscanner` on Windows.

WINDOWS

Prerequisites (Important!)

1. ****Npcap Installation Required****: Before running the scanner, you must manually install `Npcap`:
 - Download `Npcap` from: <https://npcap.com/>
 - Run the installer and follow the prompts
 - This is required for `Nmap` to function properly
2. ****Administrator Mode****: The batch file must be run as Administrator for proper functionality:
 - Right-click on `setup_scanner.bat`
 - Select "Run as administrator"
 - This is especially important for scanning privileged ports (like port 80)

Option 1: Using the GUI (Recommended)

1. Open the '`Networkscanner`' folder in your Downloads.
2. ****Right-click**** on the file named `setup_scanner.bat`
3. Select "Run as administrator"
4. Wait for the setup to finish. The web app will open in your browser at <http://127.0.0.1:5000/>
5. Use the web page to enter a target and select ports, then click 'Scan'.
 - The target field is pre-filled with `scanme.nmap.org` for testing

Option 2: Using the Command Line

1. Open Command Prompt as Administrator (Win+R, type `cmd`, press `Ctrl+Shift+Enter`).
2. Navigate to the `Networkscanner` folder:
`cd %USERPROFILE%\Downloads\Networkscanner\Networkscanner`
3. Install dependencies:
`pip install -r requirements.txt`
4. Run the app:
`python app.py`
5. Open your browser and go to <http://127.0.0.1:5000/>

TROUBLESHOOTING

Common Issues:

- ****"Nmap program was not found in path"****: Make sure `Npcap` is installed and run as Administrator
- ****Port 80 not scanning****: Run the batch file as Administrator
- ****Batch file closes immediately****: Run as Administrator and ensure `Npcap` is installed
- ****No scan results****: Check that the target is reachable and ports are open

NOTES

- The first run may take a few minutes to install Python and dependencies.
- Always run the app from inside the `Networkscanner` folder.
- The web interface is available as long as the app is running in the terminal/command prompt.
- To stop the app, close the terminal/command prompt window or press `Ctrl+C`.
- The scanner includes common ports like 21, 22, 23, 25, 53, 80, 443, etc.
- `scanme.nmap.org` is pre-filled as the default target for testing purposes. |

APPENDIX B.

Code

app.py

```
from flask import Flask, render_template, request import
os
# Set the path to nmap executable os.environ["PATH"] += os.pathsep
+ "C:\\Program Files (x86)\\Nmap" import sys

# Import the scanner and CVE lookup modules
sys.path.append(os.path.dirname(os.path.abspath(__file__)))
) from scanner import scan_host from cve_lookup import
search_cves

app = Flask(__name__)

# Recommended list of common ports for scanning
COMMON_PORTS = [
    21, # FTP
    22, # SSH
    23, # Telnet
    25, # SMTP
    53, # DNS
    80, # HTTP
    110, # POP3
    139, # NetBIOS
    143, # IMAP
    443, # HTTPS
    445, # Microsoft-DS
    3389, # RDP
    3306, # MySQL
    8080, # HTTP-alt
    5900, # VNC
    1723, # PPTP
    8000, # Common alt HTTP
    8443, # HTTPS-alt
]

@app.route('/', methods=['GET', 'POST'])
def index(): """
```

*Home page route. Displays a form to input the target and select ports to scan.
On POST, runs the scan and matches results to CVEs.*

```
"""
    scan_results = None
    cve_results = {}    if
    request.method == 'POST':
        target = request.form.get('target')
        selected_ports = request.form.getlist('ports')
        # Convert selected_ports to a comma-separated string for scanner
        ports_str = ','.join(selected_ports)    # Run the scan using
        scanner.py    scan_results = scan_host(target, ports=ports_str)
        # For each open port/service, look up CVEs    if 'scan' in
        scan_results:    for port_info in scan_results['scan']:
            service_name = port_info.get('name')    version =
            port_info.get('version')    # Only search if service name is
            available    if service_name:
                cve_results[port_info['port']] = [] # Don't call search_cves at all

        return render_template('index.html',
            target=target,
            selected_ports=selected_ports,
            common_ports=COMMON_PORTS,
            scan_results=scan_results,
            cve_results=cve_results)
        # GET request: show form with default target
        return render_template('index.html',
            target='scanme.nmap.org',
            selected_ports=[],
            common_ports=COMMON_PORTS,
            scan_results=scan_results,
            cve_results=cve_results)
@app.route('/ports') def
ports():
    return render_template('ports.html')
@app.route("/about")
def about():
    return render_template("about.html")
@app.route('/service') def
service():
    # Example service data (you can customize this)
    services = [
        {"name": "Basic Scan", "description": "Scan up to 10 ports", "price": "$9.99", "type":
"Onetime payment"},
```

```

        {"name": "Pro Scan", "description": "Scan unlimited ports", "price": "$20/month", "type":
"Membership"},
        {"name": "Enterprise", "description": "Custom solution", "price": "Contact us at 1800-
123456", "type": "Membership"},
    ]
    return render_template('service.html', services=services)
@app.route('/support', methods=['GET', 'POST']) def
support():    if request.method == 'POST':
        name = request.form.get('Name')    email = request.form.get('Email')    subject =
request.form.get('Subject')    message = request.form.get('Message')    print(f'Support
Request:\nName: {name}\nEmail: {email}\nSubject: {subject}\nMessage:
{message}')    return render_template('support.html',
message_sent=True)    return render_template('support.html')

if __name__ == '__main__':
    app.run(debug=True)

```

cve_lookup.py

import requests

Function to search for CVEs related to a given service name and version using the cve.circl.lu API

service_name: The name of the service (e.g., 'apache', 'nginx', 'openssh')

version: The version string of the service (e.g., '2.4.49') def

search_cves(service_name, version=None):

"""

Searches the cve.circl.lu API for CVEs related to the specified service and version.

Args:

service_name (str): The name of the service/software to search for.

version (str, optional): The version of the service/software.

Returns:

list: A list of dictionaries, each containing CVE details (id, summary, references, etc.).

"""

Construct the search query

if version:

query = f'{service_name} {version}'

else:

query = service_name

API endpoint for searching CVEs by keyword

url = f'https://cve.circl.lu/api/search/{query}'

```

try:
    response = requests.get(url, timeout=10)
    response.raise_for_status() # Raise an error for bad responses
    data = response.json()
    # The API returns a dictionary with a 'results' key containing a list of CVEs
    cves = data.get('results', [])
    # For each CVE, extract relevant fields
    cve_list = []
    for cve in cves:
        cve_list.append({
            'id': cve.get('id'),
            'summary': cve.get('summary'),
            'cvss': cve.get('cvss'),
            'references': cve.get('references'),
            'published': cve.get('Published'),
        })
    return cve_list
except requests.RequestException as e:
    print(f"Error querying CVE API: {e}")
return []
except Exception as e:
    print(f"Unexpected error: {e}")
return []

# Example usage (uncomment to test directly) #
if __name__ == "__main__":
    # service = "apache"
    # version = "2.4.49"
    # cves = search_cves(service, version) #
    for cve in cves:
        # print(f"{cve['id']}: {cve['summary']}")

```

scanner.py

```

import nmap
import os

# Set the path to nmap executable
os.environ["PATH"] += os.pathsep + "C:\\Program Files (x86)\\Nmap"

# Function to scan a target host for open ports and services using Nmap
# target: The IP address or hostname to scan (e.g., '192.168.1.1' or 'example.com')
# ports: (Optional) A string specifying ports to scan (e.g., '22,80,443' or '1-1024').

```

If None, scans default ports.

```
def scan_host(target, ports=None):
```

```
    """
```

Scans the specified target for open ports and service information using Nmap.

Args:

target (str): The IP address or hostname to scan. ports (str, optional): Ports to scan (e.g., '22,80,443' or '1-1024').

Returns:

dict: A dictionary containing scan results, including open ports and service details.

```
    """
```

```
    # Initialize the Nmap PortScanner object    nm =
nmap.PortScanner()    # Set the nmap path explicitly
nm.scan_command = "C:\\Program Files (x86)\\Nmap\\nmap.exe"
try:
    # Build the arguments for the scan
    scan_args = {}    if ports:
        scan_args['ports'] = ports
    # Perform the scan
    # nm.scan() returns a dictionary with scan results
    # Use simple arguments that match manual testing
    nm.scan(hosts=target, arguments="", **scan_args)
    # -sV: Service/version detection

    # Prepare results
    results = {
        'target': target,
        'scan': [] # List of open ports and their details
    }
    # nm.all_hosts() returns a list of scanned hosts (should be one in this case)
    for host in nm.all_hosts():
        # For each protocol (usually 'tcp', sometimes 'udp')
        for proto in nm[host].all_protocols():
            lport = nm[host][proto].keys() # List of ports
            for port in lport:
                port_info = nm[host][proto][port]
```

etc. # Each port_info contains details like state, name, product, version,

```
results['scan'].append({
    'port': port,
    'protocol': proto,
    'state': port_info.get('state'),
    'name': port_info.get('name'),
    'product': port_info.get('product'),
    'version': port_info.get('version'),
    'extrainfo': port_info.get('extrainfo'),
})
return results
except nmap.PortScannerError as e:
    # Handle errors (e.g., Nmap not installed, permission issues)
    print(f"Nmap scan error: {e}")
    return {'error': str(e)}
except Exception as e:
    # Handle any other exceptions
    print(f"Unexpected error: {e}")
    return {'error': str(e)}

# Example usage (uncomment to test directly) #
if __name__ == "__main__":
    # target = "scanme.nmap.org"
    # print(scan_host(target, ports="22,80,443"))

# scanme.nmap.org
```