Worm Propagation Visualization Project Documentation

1. Project Overview

This project demonstrates the propagation of two classic computer worms in a controlled and segmented virtual network environment. It includes custom worm simulation, containment strategies, monitoring systems, and visualizations to analyze the spread.

---

2. Completed Phases

Phase 1: Research and Planning

- **Selected Worms:**
    - **Code Red**
        - Year: 2001
        - CVE: CVE-2001-0500
        - Target: Microsoft IIS (Windows NT/2000)
        - Propagation: HTTP GET exploit on port 80
        - Payload: Web defacement + DDoS
    - **Voyager (Conficker-style Oracle Worm)**
        - CVE: CVE-2004-0637
        - Target: Oracle 9i/10g
        - Propagation: SQL Injection over TCP port 1521
        - Payload: DB compromise, stored procedure spread
- **Containment Strategy:**
    - VirtualBox with Vagrant
    - Host-only networks only (no NAT or bridged)
    - Segmented virtual networks with optional firewall VM (pfSense)
    - No Guest Additions or shared folders
- **High-Level Architecture Diagram:**
    - Cisco Packet Tracer used to plan topology
    - SW-web, SW-oracle, SW-bridge, SW-mon switches
    - ISR4321 router used to model Bridge1
- **Visualization Design Plan:**
    - Python + NetworkX

- Matplotlib or Plotly for graphs
- Dash/Streamlit for UI
- Optional: WebSockets for real-time

---

3. Virtual Network Architecture

Subnets

Segment	Subnet	Purpose
Web-net	192.168.56.0/24	Hosts vulnerable to Code Red
Oracle-net	192.168.57.0/24	Oracle 9i/10g vulnerable targets
Bridge-net	192.168.59.0/24	Routing/cross-segment nodes
Monitor-net	192.168.60.0/24	Monitor and control

VMs

- **Web Servers (web1–web6)**
- **Oracle DBs (oracle1–oracle5)**
- **Bridge VMs (bridge1, bridge2)**
- **Monitoring & Control (monitor, control)**

Each VM has:

- Custom script (`install_web.sh`, `install_oracle.sh`, `common.sh`)
- Assigned IP address
- Limited RAM (~256–384 MB)

---

4. Scripts and Automation

Common Provisioning Script (`common.sh`)

Includes:

- `apt update && upgrade`
- Installs: `python3`, `pip`, `git`, `vim`, `net-tools`, `curl`, `wget`, `nmap`
- Creates `/opt/wormlab` and `/vagrant/logs/`
- Prepares system for infection/monitoring tasks

Vagrant Infrastructure

- Declarative setup of 15 VMs with static IPs
- Host-only adapters to ensure containment

- Machines grouped and started incrementally
- Memory/cpu optimized for limited hardware

---

5. Monitoring and Visualization (In Progress)

- **Monitoring Script:**
    - Logs infection status per VM
    - Writes to `/vagrant/logs/`
- **Visualization:**
    - Using Python `NetworkX`, `Dash`
    - Infection status shown in real time
    - DFS propagation animation planned
    - Control panel with node inspection and reset feature

---

6. Testing and Validation (Upcoming)

- **Containment Tests:**
    - Verified IP isolation
    - No internet access on any VM
- **Functional Tests:**
    - Will simulate infection propagation
    - Log output checked for accuracy
- **System Reset:**
    - Vagrant snapshots
    - Optional: reset script to clean state

---

7. Remaining Tasks

Task	Status
Finalize monitor script	In Progress
Build visualization app	In Progress
Integrate DFS visualization	Pending
Validate propagation tracking	Pending
Record project demo video	Pending
Write final report	Pending
Reflection & AI usage summary	Pending

8. Containment Documentation

- All networks are **host-only**
- IPs limited to VirtualBox approved subnets (`/etc/vbox/networks.conf`)
- No NAT, bridged, or shared folders
- Only bridge VMs span multiple subnets
- External monitoring ensures integrity of containment

---

9. Resource Management Strategy

- Machines are launched in **small batches** (`vagrant up web1 web2`, etc.)
- Low memory allocation per VM
- Unused machines are halted (`vagrant halt`)
- Snapshots used for fast recovery after infection test

---

10. AI Usage

- ChatGPT used for:
    - Script fixing and provisioning help
    - Security containment ideas
    - Network architecture validation
    - Drafting of documentation
- All results **reviewed manually** before implementation
- The AI used **only for assistance**, not decision-making