



AmiLeaked

Product Owner: Fabian Roman

Instructor: Dr. Masoud Sadjadi

Team members:

Fabian Roman

Noah Fernandez

Austin Navas

Javier Bautista

Lucas Garcia



Introduction and Problem

- AmiLeaked is a VPN leak detector that detects leaks in an active VPN connection
- Current VPN Leak detectors do not use a baseline, so they cannot reliably detect leaks. They are also one-time based, only conducting tests once at the user's command.
- These detectors also use private databases to detect VPN associated IP's, which may not be concurrently updated or may fall out of use.





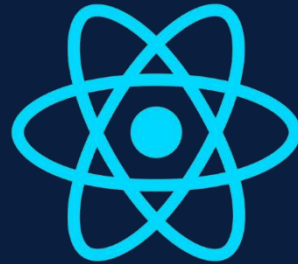
Our Solution

- A VPN leak detector that works in the background, uses a baseline, and gives the user transparency and control.
- No data is sent to third party services, and no reliance on external databases



Languages and Resources used

- React
- JavaScript
- HTML
- CSS



- VSCode
- Chrome
- Node.js
- Vite





Implementation Highlights

Local Multi-vector Leak Detection

- No data is sent to third party services, and no reliance on external databases

Supports a wide variety of platforms and devices

- Works on Chrome Manifest V3
- Extension format allows for open platform support



Project Development

Sprint 1:

- Develop knowledge about public IP observation and safe detection methods
- Research IP vectors associated with VPNs

Sprint 2:

- Develop simple script to fetch public IP using ipify API
- Prototype WebRTC detection method using STUN servers
- Research false positive leak detection scenarios

Sprint 3:

- Evaluate browser specific constraints
- Determine initial leak type to implement
- Prototype periodic IP capture based on a time interval





Project Development

Sprint 4:

- Create GitHub repo and initialize Vite project
- Implement IP fetch logic in the project
- Learn safe branching methods for GitHub

Sprint 5:

- Continue project development, finished IP detection for all vectors
- Create initial UI
- Initialize baseline capture logic

Sprint 6:

- Error handling, rigorous testing
- Polish UI, implement background scan logic
- Rigorous testing and determining user scenarios

Sprint 7:

- Completed baseline comparison and UI features
- Final error handling





Challenges and Lessons learned

WebRTC limitations:

- STUN servers were not able to be used by Chromes background service worker by default
- WebRTC IP was not being captured in background detections as a result
- Fixed by adding secondary offscreen JS module

Scrum framework is tough but reliable:

- Team members were not able to keep up at first
- Keeping up with tasks became easier with time due to the nature of Scrum

Forced to hide non-working features

- Features such as OS notifications and auto-start background scan were hidden due to functionality complications.





Summary and Future work

- Our team was able to develop a robust VPN leak detector app that goes beyond what typical leak detectors do. It features a non-invasive badge notification system, and high-accuracy leak detection results.
- Over the course of the first 3 sprints, extensive research was done by all team members to gain a good understanding of current leak detectors and IP vector information. Development was up and running beyond that, which allowed us to finish the project without delays.
- Future work includes fixing the hidden features, and expanding on the functionality of the application and its settings. Alternative IP fetching planned in order to remove reliance from 3rd party API's

