Contents

Plan

Challenges

Summary:

- Stability issues
  Certain issues are known such as network host name discovery. We will develop test cases and a tool to resolve these. Additionally we will add some tool resource health information to see packet loss, packet details options and CPU/memory use. This will also result in some research into how other NodeJS projects handle resource exhaustion.
- Platform changes
  Prior code uses OSX/Linux app. We could move this onto a RaspberryPi greatly simplifying setup and installation but at the cost of less usability for a wider user base. So for at least the Alpha we will stick with App, and add windows support.
- Installation simplicity
  Currently network topology requires some manual configuration which we can remove in Alpha.
- Windows
  Windows packet capturing in NodeJS is less mature. Additionally both Windows and OSX currently need Root to setup network. We will explore using a installer to setup network first. Alternatively we will investigate creating a Linux bootable image. Bootable Linux images not support by many OSX laptops though. This will be investigated.
- Security issues
  A conversation should be had about the risks to trainers that load and run the software. Should they be advised to run it on a seperate host or should we go through the effort to develop a live CD.

For additional detailed discussion on challenges see **Challenges Details** section.

METHODOLOGY

Alpha

Tasks

- Create github repo
- Windows version
  - Test use of windows VM
  - investigate and implement pcap
  - investigate installer
  - port existing code to Windows
- Improve installation (OSX + Windows)
  - Installer or simplified configuration
  - Investigate ability to monitor network cards without root
- Build test cases and test tools (e.g. Name discovery testing)
  - Consider testing reactjs app framework for this https://github.com/electron/electron-packager/issues/33
  - Consider testing installer with this tasks
- Include additional status information to UI (CPU utilization, memory, nodejs specific resource limits, packet loss)
- Plan UI improvements for Beta

Beta

Tasks:

- Unify and simplify snifferjs and always_on code into one executable
- Implement plugin architecture
- Attempt to implement OS detect

RC1

Tasks:

- Develop stretch goals:
  - SSL stripping
  - Wifi network history extraction (pinapple integration)
  - Zeroknowledge thret sharing and discovery

Challenges Details

- Stability issues
- Getting names of network hosts has been hit and miss. Will need to explore this in more depth
- In some cases students said they accessed a site on their device but it did not show up in list. This should be investigated. Because it may be hard to replicate deeper inspection tools should be provided. This will likely include investigating NodeJS async loop allocation monitoring.
- Inspection tools and information on resource drain (CPU, power, memory) and packet loss should be easily available
- Currently the MiTM server serves as primary gateway to clients. This is done by turning off DHCP on the wifi router and providing DHCP and DNS locally. We should find an alternative solution. Options:

  1. Server as bridge between WiFi's,
     no router configuration,
     issues with device name discovery:

     ```
     wan:UpstreamWiFi:lan
                       |
     eth0:MiTM-Server:wlan0
      |
     wan:CheckDeviceWifi:lan
                          |
                         clients...
     ```

     This topology requires no configuration changes on wifi routers. However, MiTM-Server can no longer send broadcast mDNS name discovery packets to clients.

  2. Server as bridge between WiFi's,
     turn off DHCP on router,
     Server provides DHCP and is default gateway,
     Server connects to LAN of WiFi to broadcast for device names:

     ```
     wan:UpstreamWifi:lan
                       |
     eth0:MiTM-Server:wlan0
       |
     _lan_:CheckDeviceWifi
       |
     clients...
     ```

     This is the topology I've used until now. MiTM-Server and Clients all connect to same LAN. MiTM-Server provides DHCP and DNS. Requires during off DHCP on CheckDeviceWifi.

3. Server as bridge between WiFi's,
   no router configuration,
   Server has second eth (via USB),
   Server connects to both wan and lan,

   ```
    wan:UpstreamWifi:lan
                       |
    eth0:MiTM-Server:wlan0
      |   MiTM-Server:eth1
      |                  |
   wan:CheckDeviceWifi:lan
                         |
                       clients...
   ```

   This topology requires no change to WiFi routers but does require using a second network card so that the server can set network broadcast packets for name discovery. This will result in different code and additional testing.

- Platform changes
- Prior software built as OSX/Linux app. Changing to a application box (on Raspberry Pi) would provide seamless setup, lower security risk, but be more difficult to update or require a update module that needs to be built, tested and may add other trust risks.
- We will for Alpha version stick with app, extending to Windows and adding some installation and setup improvements. It would not be difficult to later switch this software to a Raspberry Pi. I'm also inclined to stick with app for later versions though as I think this will result in wider community use and longer project longevity.
- Installation simplicity
- to see all network traffic workstation with MiTM software needs to either be the primary network gateway or be connected to a network hub with the gateway so that it can see all traffic.
- Windows
- libpcap support on Windows is very different from Linux. The node_pcap library will not work. There is an alternative less tested cap library we can likely utilize.
- Security issues
- It may be important to consider that the MiTM software will not be heavily maintained and so security of trainer laptops could be a concern. Affects:

  1. may require dedicated laptops
  2. may need to explore feasibility of packet capture within VM
  3. may need explore making bootable linux image

- Startup requires root to obtain control of network interface. Currently code reduced privileges after setting up network interface. We might explore:

  1. Installer that changes network permissions once
  2. using bootable linux image

TIMELINE:

Gant date/milestones chart:
https://pad.ano.la/sheet/#/2/sheet/edit/-8373R6YBZ8BVFaWe5fEDTV4/

- Tech Plan - schedule (3 days)
  - schedule
  - tech stack
    * network setup
      * router, rpi/nvidia, phone, laptop?
      * Rpi Advantage: w/LCD less setup time (router in one object)
      * Rpi Advantage: runs on all platforms, no installation app
      * Rpi Disadvantage: Harder to update (or have to build update element)
      * Reactjs advantage: phone sharing with upstream gsm
      * Reactjs disadvantage: limited hardware? (client limit for android sharing: 3-12 or need workaround) requires rooted device

      https://android.stackexchange.com/questions/186487/increase-maximum-number-of-connection-of-wi-fi-h
      * Docker?
      * windows?
      * can eth in promiscuous mode?
    * windows dev
      * via VM https://www.electronjs.org/docs/tutorial/development-environment use free windows VM's https://developer.microsoft.com/en-us/windows/downloads/virtual-machines/
      * still need something for testing. might as well get something we can develop on as well
      $400 ThinkPad W541 w/nvidia N15P-Q1 2 GB
      $850 Yoga 730 w/nvidia GTX 1050 4 GB or w/nvidia MX250
      those are not the smallest. if you can give up nvidia you can get a x1 gen 2 for $500
      cheapest for AI:
      https://favouriteblog.com/best-laptops-for-deep-learning/
      * windows support for node_pcap

      https://github.com/node-pcap/node_pcap/search?q=windows&unscoped_q=windows

https://github.com/node-pcap/node_pcap/issues/249 Jan 7, 2019

https://github.com/node-pcap/node_pcap/issues/249#issuecomment-586201286
> This module has never supported (and can't easily support) windows.
> That's because on windows you can't get a pollable socket:
> pcap_get_selectable_fd() is not available on Windows.
> You can work around this by doing blocking reads in an additional thread, which is what cap does. This could be merged back into pcap at some point.
https://github.com/node-pcap/node_pcap/pull/98
> Tested on Windows 7, Could compile in release mode with winpcap developer kit. Need to use wpcap.lib instead of pcap.lib.
* windows alternative cap js
https://github.com/mscdex/cap
https://github.com/mscdex/cap/issues/95
so works better on windows perhaps, but not as active as node_pcap
Send email to issue author asking their opinion
* windows electronjs support?
* root perms
https://github.com/electron/electron-packager/issues/87
> I know this is a corner case, but our app (ixmaps) uses raw-sockets so needs to run as root
https://github.com/ixmaps/IXmapsClient
> "IXmapsClient needs to be executed in a terminal with administrator's privileges. For this reason, when double clicking IXmapsClient-Shortcut, a new terminal window will be opened asking permission to run the application as an administrator; enter your admin password to proceed."
> That sounds like something an installer (#33) would handle.
https://github.com/electron/electron-packager/issues/33
options:
https://github.com/electron-userland/electron-builder

https://stackoverflow.com/questions/54652002/how-can-i-take-persistent-permissions-in-electron-app
You need to run the applicaton with administrator privilege. If you are using electron builder to build the app, use the requestedExecutionLevel value as "requireAdministrator"
https://www.electron.build/configuration/win

  * other:
    * electronjs
    * reactjs: https://proton-native.js.org/#/
  * plugin arch
  * os detect
- challenges
  * packet loss info
  * reproduce traffic

    see reply issues in cap github, or node_pcap
    * pcap replay
      https://www.colasoft.com/packet_player/
      https://github.com/appneta/tcpreplay
      https://metron.apache.org/current-book/metron-deployment/ansible/roles/pcap_replay/index.html

- UI: Stakeholder input (2 days)
  - inquiry with stakeholders
- Alpha (11 days)
  - Windows support
    * libpcap
    * setup electron if time
  - first UI changes
  - simpler install
  - simpler network
  - session save/restart

  - Improvements
    * name discovery types and records
    * test cases for: dns server self, dns server router, dns server 8.8.8.8
    * test cases for: when we are not default gw (broadcast names?)
    * tools: show dns sniff (shell script or pcap)
- Demo? (5 days)
- RE-ASSES:
  - UI changes
  - schedule changes

  - decide if we have the time for stretch goals, order pinapple etc
- Beta (11.5 days)
  - electron app win+osx
  - hidden install+setup
  - new features
    * plugin
    * os detect
- Demo (3 days)
- RC1 (14.5 days)
  - testing
  - bug fixing

  - stretch goals
    * wifi ssid extract
    * sslstrip
    * 0knowledge discovery

- User documentation (3 days)
  distributed among other items

- Source code published (2 days)
  distributed among other items