‘Safe’ network analysis
Generating network traffic captures within a virtual network.

Presented by Andrew Martin
Introduction

- What is a sniffer
- How does sniffing work
- Usages
- Scenarios
- Building safe repositories using VM technologies
- Wireshark
- Movie
What is a sniffer

- Sniffer is a term given to applications that capture network data
- Some examples include:
  - Wireshark
  - Snort
  - EtherApe
How does sniffing work?

- Data enters the network
- NICs are supposed to be honest and turn away packets not meant for them
- Sniffing applications simply tell the NIC card to lie
Applications

- Both legitimate and illegitimate purposes
- End users complaining of network “problems”
- User cannot connect to a machine
- Curiosity over one’s instant messenger conversation
- Need access to a system to which you currently have no access
Problem

- Educate students about network sniffing technologies
- Production networks generate massive amounts of traffic
  - Realtime analysis is impractical
- Protect privacy of network users
Situation

- Capture data and store it for later educational analysis
- AND keep network users’ data private?
Solution

- Solution 1: Capture traffic and analyze it later
  - Chances are private data will be captured
- Solution 2: Capture traffic on a non-production network
  - Costly to create a non-production network
- Solution 3: Create virtual network and capture traffic
  - Ding ding ding, we have a winner!
Virtualization

- It's not new
  - Been around since the 1960s
- Cheap
- Can run off a fairly low-end PC
- One PC can host a slew of VM
- Create a heterogeneous virtual lab with just one PC
Virtual Lab Setup

- Host: VMWare Workstation
- FreeBSD - extensive collection of software apps
  - security utilities easily installed via port
- Red Hat 7.3 & Windows 2000
  - relatively old and susceptible to network attacks
Virtual Lab Setup Cont’d

- Network is a virtual network controlled by VMWare workstation
- Uses private addresses
- Connected to the outside world via using N.A.T.
Capturing network traffic

- Build a repository consisting of two types of traffic
  - Normal
    - FTP, HTTP, SMTP, IRC, SSH...
  - Irregular
    - Network scans, exploits, infected computers
    - using tools such as nmap, or metasploit framework
Personal Favorite

- Wireshark (formally Ethereal)
- Network traffic capturer and analyzer
- Uses libpcap (or winpcap) library to abstract network types and support many more networks
- Intuitive interface
- Supports capture and display filters
Useful Techniques

- ARP Spoofing / Poisoning Detection
  - Attacker will try to trick a target into “thinking” that the attacker is not who they say they are
  - Wireshark can easily detect such attacks
    - ARP -- simple filter
  - tshark can be scripted to automatically capture traffic
    - scripts can be written to parse data to look for certain types of network data - lua.org
Getting what you want

- There are many filters that can be applied
  - network - net 192.168.1.0/24
  - source - src 192.168.1.15
  - destination - dst 192.168.1.10
  - host - host 192.168.1.1
- Combining filters is quite useful
  - dst port 135 and tcp port 135 and ip[2:2]==48 (blaster worm)
Display filters

- Capture filters and display filters sometimes have different reserved words
- Display filters have a nice front end for assistance
TCP Streams

- Follow TCP stream
  - Wireshark will display the application layer data in the order in which it was received
Packet reassembly

- Save packets that contain binary data, and fuse them together
Conclusion

- Privacy is incredibly important while educating students on the importance of network analyzation
- Best to generate samples on a private network
- Virtual networks are much cheaper than physical networks
- Wireshark in the hands of a skillful user is both powerful and dangerous
References

- http://www.wireshark.org/docs/wsug_html_chunked/
- http://wiki.wireshark.org/TCP_Reassembly
- http://www.wiresharktraining.com/
- http://www.vmware.com/
- http://www.youtube.com/watch?v=7ezGTP99xSw