Computer Network Defense

we begin with the triad
- patch updates
  - #1 most overlooked security technique
  - as vulnerabilities are found, patches are released
  - how long (on average) do you think a vulnerability exists before it is discovered?
    - 342-ish days (uhhh, yeah, almost one year!)
  - patches may make you safe from *most* attacks
    - just not 0-day attacks

- malware protection
  - no, you are not invincible!
    - no matter what you think

  - malware?
    - viruses
    - worms
    - bacteria
    - trojans
    - rootkits
    - spyware (e.g., sniffers, keyloggers, etc)
    - adware (e.g., annoying popups, spam, phishing, etc)

  - defense?
    - anti-virus
    - anti-spam/anti-adware
    - anti-malware?
      - hash detection, basically

- firewall
  - take care of what's on your system
  - two philosophies
    - block based on port (which usually ties to services/protocols)
      - any application attempting connection on a port is blocked
      - this is the Linux way
    - block based on application
      - a single application is blocked
      - this is the Windows way
      - usually means having to interact more with the firewall
        - which is usually annoying

  - stateful vs. stateless
    - do we treat each packet uniquely (no past memory)? → stateless
    - or do we use the past to infer something about the now? → stateful

  - h/w (e.g., router, dedicated Linux firewall) vs. s/w (e.g., zone alarm, windows firewall, etc)
    - you can setup a Linux box as the front facing thing in your network
    - setup iptables/netfilter properly
      - iptables: the Linux interface to netfilter
      - netfilter: the core Linux firewall
      - any downloaded firewall simply makes interacting with iptables more “friendly”

defensive operations

what can we do to “protect” ourselves?

one option is to encapsulate our services/OS

e.g., virtualization (hypervisor and virtual machines)

e.g., virtualbox, vmware, xen, proxmox, etc

e.g., chroot jails (**see the relevant document on the class web site**)

e.g., docker containers

defense in depth

don't depend on a single mechanism for protection

layered approach (multiple layers of defense) – like an onion in your network

idea: use several varying methods

sometimes, we delay rather than prevent: yield space in order to buy time

so it should prevent security breaches while giving time to respond

defense in breadth

there are many attack vectors (i.e., just having a firewall won't guarantee security)

we must try to cover all attack vectors

IDS/IPS

how can we detect intrusions?

how can we detect attackers?

could we protect/prevent in addition to detect?

these usually inspect packets (sometimes deeply)

tcp wrappers (rules)

maybe we can think about this being like a filter for tcp packets

we can scan, log, anonymize, etc

and maybe we could detect/protect/prevent via tcp wrappers

PDR^3 (or should it be PDRER?)

prevent

we're a “pill” society

we prefer to take care of the symptoms, not the cause

and that's often a bad idea (but a money-making one!)

better idea: identify the cause and prevent the problem from occurring again

but that takes work (effort) – that's why we're a symptom-based society

so best case is to prevent security breaches and vulnerability exploits

but that's not always possible, particularly in cyberspace

detect

if we can't prevent, we must find out when we have a problem

so use an ids, ips, idps

and also firewall, patches, anti-virus (i.e., the triad)

respond

if we detect, we can't just let something bad happen

what to do, what to do?!

how proactive can we be?

do we just secure our system and repair?

then prevent the perpetrator from doing it again (how?)

can we “engage?”

can we find out who did this and where they live?

recover
if our system was compromised, we may need to recover
how might we do this?
or might we endure instead of recover? or both?

restore
maybe our system is irrecoverable
so we take this as a learning experience
we restore from some previous backup
then we look at how to prevent this from happening again
and we loop back to the beginning...

avoid?
how the hell do we do this?
threat avoidance
threats simply don't matter
we don't care about detection, mitigation, prevention, attribution
we have an invisibility cloak
e.g., beaconing malware, unauthorized network users/apps, port knocking

honeypots prove useful
they have no production value
they lure attackers
we want to know what they do, what they use, how they do it
honeynets

interesting and relevant types of attacks that we may have to defend against
ddos (the holy grail)
dos: denial of service attack
attempt to make computer resources unavailable
ddos: originates from multiple systems
how?
consume computer resources (bandwidth, cpu, disk space)
disrupt configuration information (e.g. routing information)
disrupt state information (e.g. reset tcp sessions)
disrupt physical network components
obstruct communication

botnets
a bunch of zombies!
software agents that run autonomously and automatically
mostly interpreted to be malicious; but can be legitimate (e.g., SETI)
compromised via
drive-by-downloads (RTFM!)
awareness is important (in everything actually)
browser exploits
worms
trojans
backdoors
bot herder/master establishes C3

- often takes place on irc server
- usually runs hidden in a covert channel
- Dutch police once found a 1.5 million node botnet!
  - they are now larger!
- used in many ways and typically auctioned to highest bidder; for:
  - spam, ddos, click fraud, adware, spyware, etc

- script kiddies
  - those who use scripts or programs developed by others to attack computer systems
    - but they really don't know anything more than that
    - no knowledge of the underlying concepts
    - they're just annoying
  - most “hackers” are actually script kiddies

offensive operations
- sometimes, the best defense is knowledge of the offensive tactics
- so many of the things here are often employed defensively
  - let me see what I can gather from my own networks and systems from an offensive side
  - so that I can build a better defensive side

- reconnaissance and footprinting
  - useful to see if we might want to gain access to a system we don't have access to
  - recon: what's there? what systems exist?
  - footprinting: what specific things can we gather about those systems?
  - we might want to know a few things about a system:
    - what OS it runs
    - what hardware it has
    - what servers are running and on what ports
    - including versions of all of these (some may have known vulnerabilities!)

  - recon tools
    - nmap: security scanner for network exploration and security audits
    - nemesis: packet crafter and sender
    - python-scapy: packet swiss army knife
    - netcat: tcp swiss army knife
    - telnet: not as good as netcat

  - recon/footprinting tactics
    - port scanning
      - probes remote host for open ports
      - used to verify security policies and identify running services
      - portscan: scan for listening ports
      - portsweep: scan multiple hosts for a specific port
        - some worm may portsweep many hosts for a single port (vulnerability)
      - port status
        - open/accepted: something is listening
        - closed/denied/not listening: connection is denied
        - filtered/dropped/blocked: no reply (firewall?)

    - tcp scanning
      - use OS network functions
      - in nmap, called a connect scan

- on connect, handshake performed and connection closed
- no special privileges required
- no low level control

syn scanning
- uses raw ip packets and monitors for responses
- known as half-open scanning because never actually opens a full TCP connection
- port scanner generates a SYN packet
- if target port is open, host responds with SYN-ACK
- port scanner responds with RST and closes connection before handshake
- we can get many details this way
- target service never actually receives a connection
- usually requires privileges

udp scanning
- udp is a connectionless protocol
- response comes only if a port is closed
- so absence of response implies port is open
- most scanners use this method
- firewalls can fool scanner

network sniffing (particularly under the same subnet) – "sniffer"
- packet analysis
- intercepts/logs network traffic (packets)
- we can then decode/analyze these packets
- uses:
  - analyze network problems
  - detect network intrusion attempts
  - gain info for possible network intrusion
  - monitor network usage
  - gather/report network stats
  - filter content from traffic
  - spy on users/collect sensitive information
  - reverse engineer proprietary protocols
  - debug client/server communication
  - debug network protocols

- tools
  - tcpdump, wireshark (formerly ethereal)
  - python-scapy (wrap your own sniffer around it)

arp spoofing
- arp = address resolution protocol
- can be used to poison (arp poisoning)
- man-in-the-middle
  - can stop traffic
  - can modify traffic
- can only be used on networks that make use of arp
- tools
  - ettercap