WILLIAM & MARY
CHARTERED 1693

CSCI 667: Concepts of Computer Security

Prof. Adwait Nadkarni

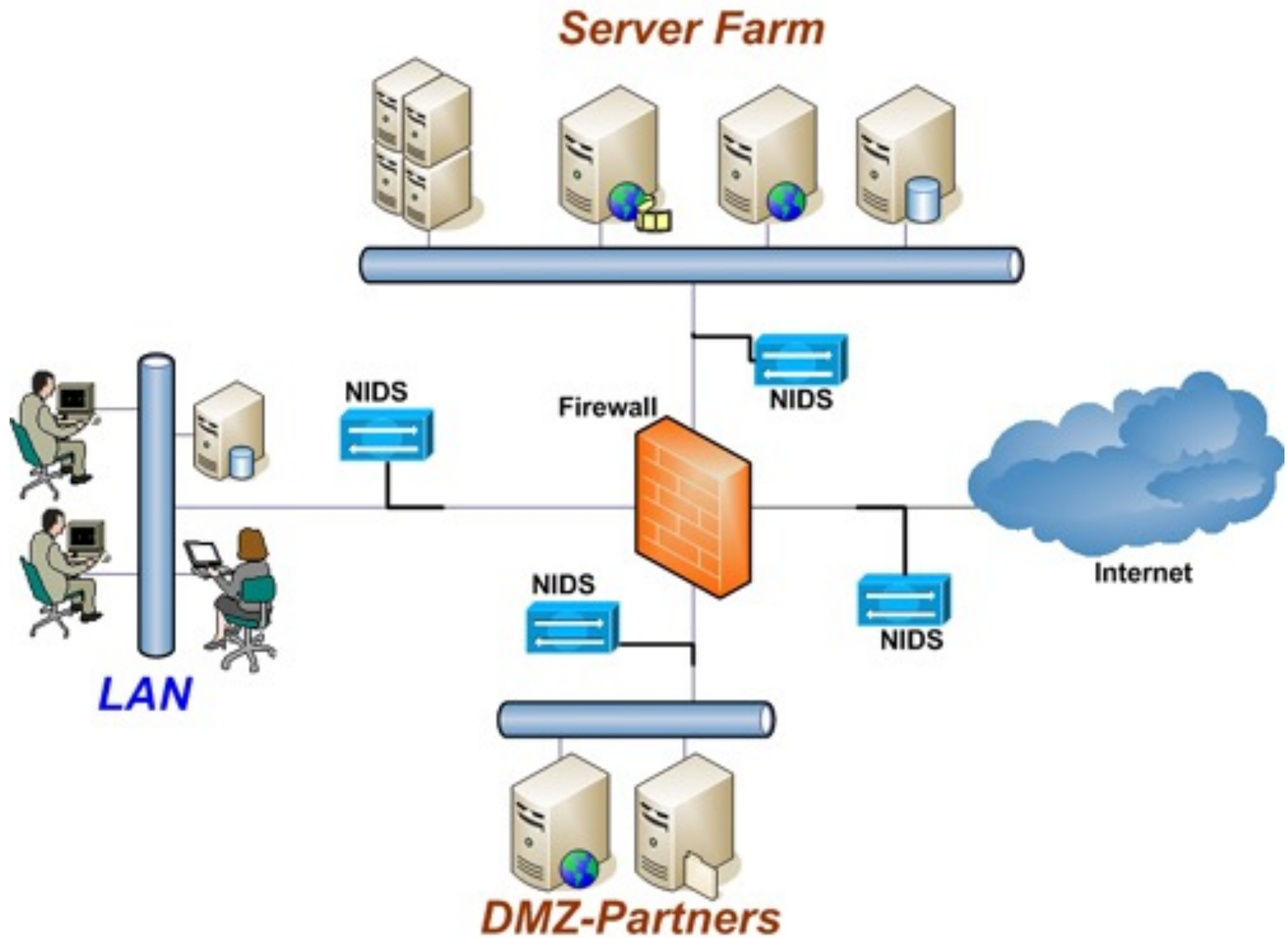
Derived from slides by William Enck, Micah Sherr, and Patrick McDaniel

Network Intrusion Detection Systems (NIDS)

Intrusion Detection Systems

- **Authorized eavesdropper** that listens in on network traffic
- Makes determination whether **traffic contains malware**
  - usually compares payload to virus/worm signatures
  - usually looks at only incoming traffic
- If malware is detected, IDS somehow raises an alert
- Intrusion detection is a **classification problem**

Example Setup



Detection via Signatures

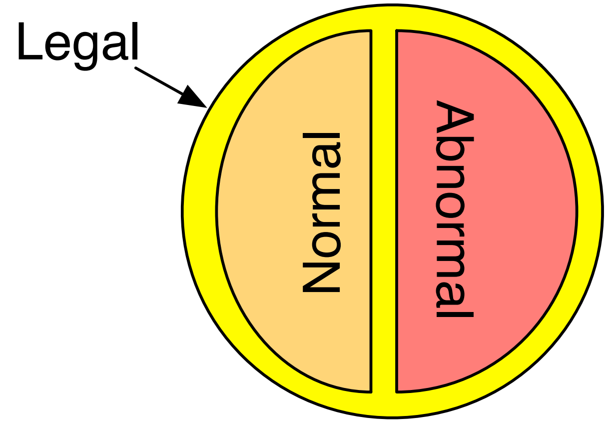
- Signature checking
  - does packet match some signature
    - suspicious headers
    - suspicious payload (e.g., shellcode)
  - great at matching known signatures
  - Low *false positive* rate: **Q: WHY?**
  - Problem: not so great for zero-day attacks -- **Q: WHY?**

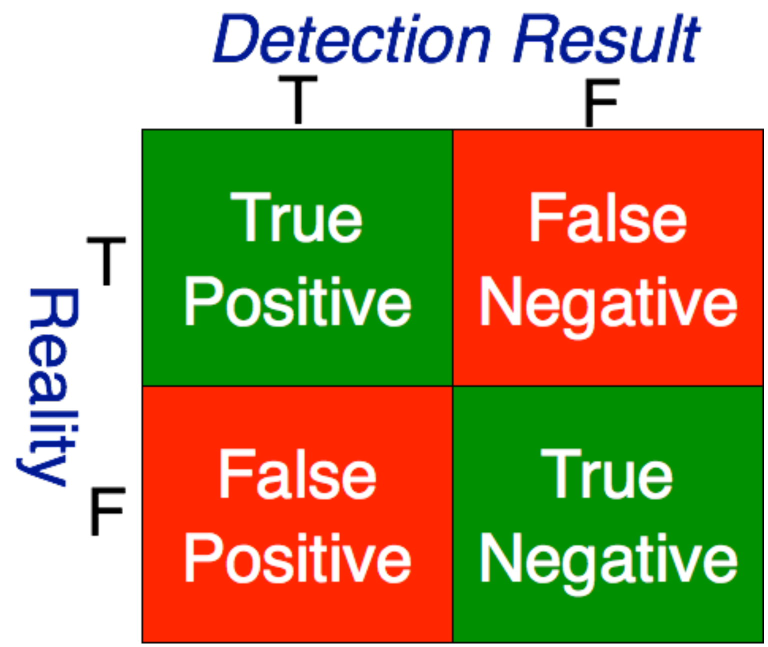
Anomaly Detection

- *Learn what "normal" looks like.*
- Frequently uses ML techniques to identify malware
- Underlying assumption: malware will look different from non-malware
- **Supervised learning**
  - IDS requires learning phase in which operator provides pre-classified ***training data*** to learn patterns
  - {good, 80, "GET", "/", "Firefox"}
  - {bad, 80, "POST", "/php-shell.php?cmd='rm -rf /'", "Evil Browser"}
  - ML technique builds model for classifying never-before-seen packets
  - Problem: *False Learning*
  - Problem: is new malware going to look like training malware?

Confusion Matrix

- What constitutes an intrusion/anomaly is really just a matter of definition
  - A system can exhibit all sorts of behavior
- Quality determined by the consistency with a given definition
  - Context-sensitive (i.e., what is “positive/true”?)





Metrics

- **True positives** (TP): number of correct classifications of malware
- **True negatives** (TN): number of correct classifications of non-malware
- **False positives** (FP): number of incorrect classifications of non-malware as malware
- **False negatives** (FN): number of incorrect classifications of malware as non-malware

Metrics

(from perspective of detector)

		Detection Result	
		T	F
Reality	T	True Positive	False Negative
	F	False Positive	True Negative

- **False positive rate**:

$$FPR = \frac{FP}{FP + TN} = \frac{\text{\# benign marked as malicious}}{\text{total benign}}$$

- **True negative rate**:

$$TNR = 1 - FPR = \frac{TN}{FP + TN} = \frac{\text{\# benign unmarked}}{\text{total benign}}$$

- **False negative rate:**

$$FNR = \frac{FN}{FN + TP} = \frac{\text{\# malicious not marked}}{\text{total malicious}}$$

- **True positive rate:**

$$TPR = 1 - FNR = \frac{TP}{FN + TP} = \frac{\text{\# malicious correctly marked}}{\text{total malicious}}$$

Base Rate Fallacy

- Occurs when we assess P(X|Y) without considering prior probability of X and the total probability of Y
- Example:
  - *Base rate* of malware is 1 packet in a 10,000
  - Intrusion detection system is 99% accurate (given known samples)
    - 1% false positive rate (benign marked as malicious 1% of the time)
    - 1% false negative rate (malicious marked as benign 1% of the time)
  - Packet X is marked by the NIDS as malware. *What is the probability that packet X actually is malware?*
    - Let's call this the "true alarm rate," because it is the rate at which the raised alarm is actually true.

Bayes' Rule

- Pr(*x*) function, probability of event *x*
  - Pr(sunny) = .8 (80% of sunny day)
- Pr(x|y), probability of x given y
  - Conditional probability
  - Pr(cavity|toothache) = .6
    - 60% chance of cavity given you have a toothache
- Bayes' Rule (of conditional probability)

$$Pr(B|A) = \frac{Pr(A|B) \cdot Pr(B)}{Pr(A)}$$

- Assume: Pr(cavity) = .5, Pr(toothache) = .1
- What is Pr(toothache|cavity)?

Base Rate Fallacy

- How do we find the true alarm rate? [i.e., Pr(IsMalware|MarkedAsMalware)]

$$Pr(IsMalware|MarkedAsMalware) = \frac{Pr(MarkedAsMalware|IsMalware) \cdot Pr(IsMalware)}{Pr(MarkedAsMalware)}$$

- We know:
  - 1% false positive rate (benign marked as malicious 1% of the time); TNR= 99%
  - 1% false negative rate (malicious marked as benign 1% of the time); TPR= 99%
  - *Base rate* of malware is 1 packet in 10,000
- What is?
  - Pr(MarkedAsMalware|IsMalware) = $\boxed{\text{TPR = 0.99}}$
  - Pr(IsMalware) = $\boxed{\text{Base rate = 0.0001}}$
  - Pr(MarkedAsMalware) = ?

$$Pr(MarkedAsMalware|IsMalware)$$
$$= \frac{\text{\# malicious correctly marked}}{\text{total malicious}}$$
$$= \frac{TP}{FN + TP} = TPR$$

Base Rate Fallacy

- How do we find Pr(MarkedAsMalware)?

$\boxed{= \Pr(\text{MarkedAsMalware}|\text{IsMalware})\Pr(\text{IsMalware}) + \Pr(\text{MarkedAsMalware}|\text{IsNotMalware})\Pr(\text{IsNotMalware})}$

- So what is?
  - Pr(IsMalware) = base rate = 0.0001
  - Pr(IsNotMalware) = $\boxed{1 - \Pr(\text{IsMalware}) = 0.999}$
  - Pr(MarkedAsMalware|IsMalware) = TPR = 0.99
  - Pr(MarkedAsMalware|IsNotMalware) = $\boxed{\text{FPR} = 0.01}$

Pr(A|!B) = 1 – Pr(!A|!B)
Pr(A|B) = 1 – Pr(!A|B)

$$\boxed{\begin{aligned} &Pr(MarkedAsMalware|IsNotMalware) \\ &= \frac{\#\text{ benign marked as malicious}}{\text{total benign}} \\ &= \frac{FP}{FP+TN} = FPR \end{aligned}}$$

- So Pr(MarkedAsMalware) = 0.99 * 0.0001 + 0.01 * 0.9999 ~= 0.01

Base Rate Fallacy

- How do we find the true alarm rate? [i.e., Pr(IsMalware|MarkedAsMalware)]

$$Pr(IsMalware|MarkedAsMalware) = \frac{Pr(MarkedAsMalware|IsMalware) \cdot Pr(IsMalware)}{Pr(MarkedAsMalware)}$$

$$= \frac{0.99 \cdot 0.0001}{0.01} = 0.0099$$

- Therefore *only about 1% of alarms are actually malware!*
  - What does this mean for network administrators?

Base Rate Fallacy

(summary)

- Let Pr(M) be the probability that a packet is actually malware (the base rate)
- Let Pr(A) be the probability that that the IDS raises an alarm (unknown)
- Assume we also know for the IDS
  - Pr(A|M) = TPR = 1 - FNR
  - Pr(A|!M) = FPR
- Then the true alarm rate is

$$Pr(M|A) = \frac{Pr(A|M) \cdot Pr(M)}{Pr(A|M) \cdot Pr(M) + Pr(A|!M) \cdot Pr(!M)}$$

- **Note the strong influence of Pr(M)**

Where is Anomaly Detection Useful?

System	Intrusion Density P(M)	Detector Alarm Pr(A)	Detector Accuracy Pr(A|M)	True Alarm Rate P(M|A)
A	0.1		0.65	
B	0.001		0.99	
C	0.1		0.99	
D	0.00001		0.99999	

$$\Pr(B|A) = \frac{\Pr(A|B)\,\Pr(B)}{\Pr(A)}$$

Where is Anomaly Detection Useful?

System	Intrusion Density P(M)	Detector Alarm Pr(A)	Detector Accuracy Pr(A|M)	True Alarm Rate P(M|A)
A	0.1	0.38	0.65	0.171
B	0.001	0.01098	0.99	0.090164
C	0.1	0.108	0.99	0.911667
D	0.00001	0.00002	0.99999	0.5

$$\Pr(B|A) = \frac{\Pr(A|B)\,\Pr(B)}{\Pr(A)}$$

Base-rate Fallacy in the real world

Health Nerd
@GidMK

Follow

So, according to this, the false positive rate for the Apple Watch in detecting atrial fibrillation is 0.04% (99.6% correct)

This means that, on average, Apple Watches will be wrong more than 80% of the time

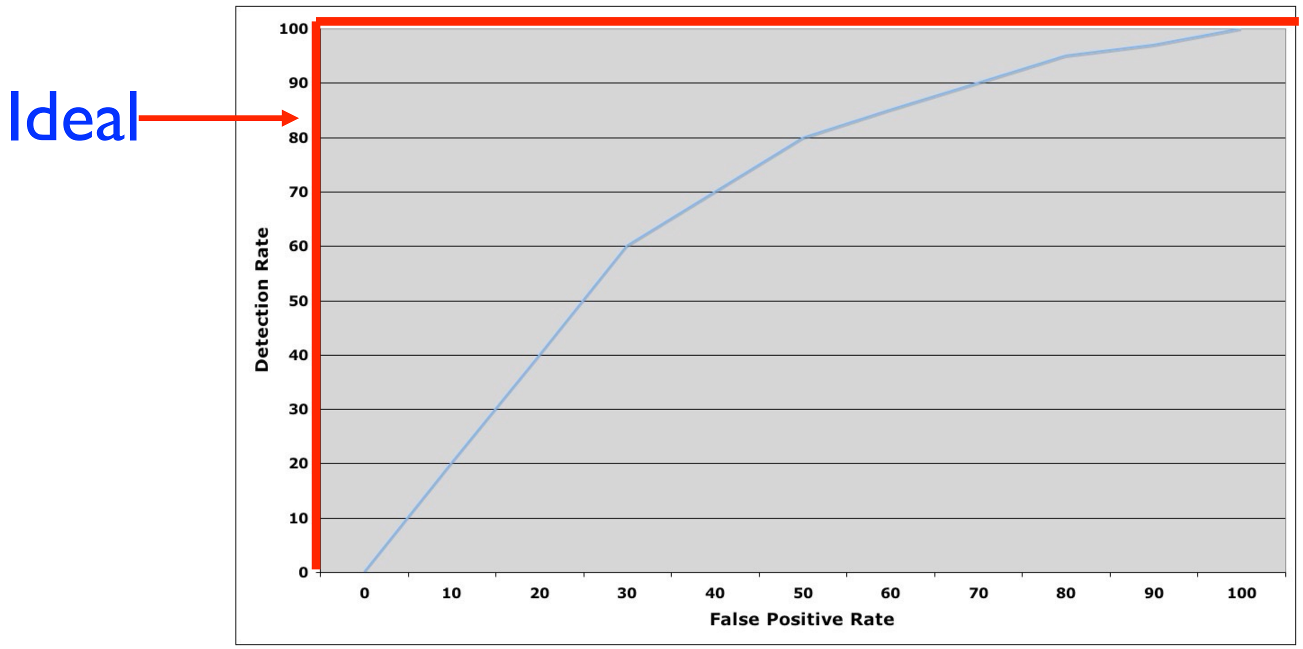
Sound counterintuitive? This is the issue with population screening

> **STAT** @statnews
> Apple submitted two studies to FDA to get clearance for the new Apple Watch EKG app. Here's the data. buff.ly/2QuhGmG

5:28 AM - 14 Sep 2018

The ROC curve

- Receiver Operating Characteristic (ROC)
  - Curve that shows that detection/false positive ratio (for a binary classifier system as its discrimination threshold is varied)



- Axelsson talks about the real problem with some authority and shows how this is not unique to CS
  - Medical, criminology (think super-bowl), financial

Example ROC Curve

- You are told to design an intrusion detection algorithm that identifies vulnerabilities by solely looking at transaction length, i.e., the algorithm uses a packet length threshold T that determines when a packet is marked as an attack (i.e., less than or equal to length T). More formally, the algorithm is defined:

$$\mathrm{D}(\mathrm{k},\mathrm{T}) \rightarrow [0,1]$$

- where k is the packet length of a suspect packet in bytes, T is the length threshold, and (0,1) indicate that packet should or should not be marked as an attack, respectively. You are given the following data to use to design the algorithm.

  attack packet lengths: 1, 1, 2, 3, 5, 8

  non-attack packet lengths: 2, 2, 4, 6, 6, 7, 8, 9

- Draw the ROC curve.

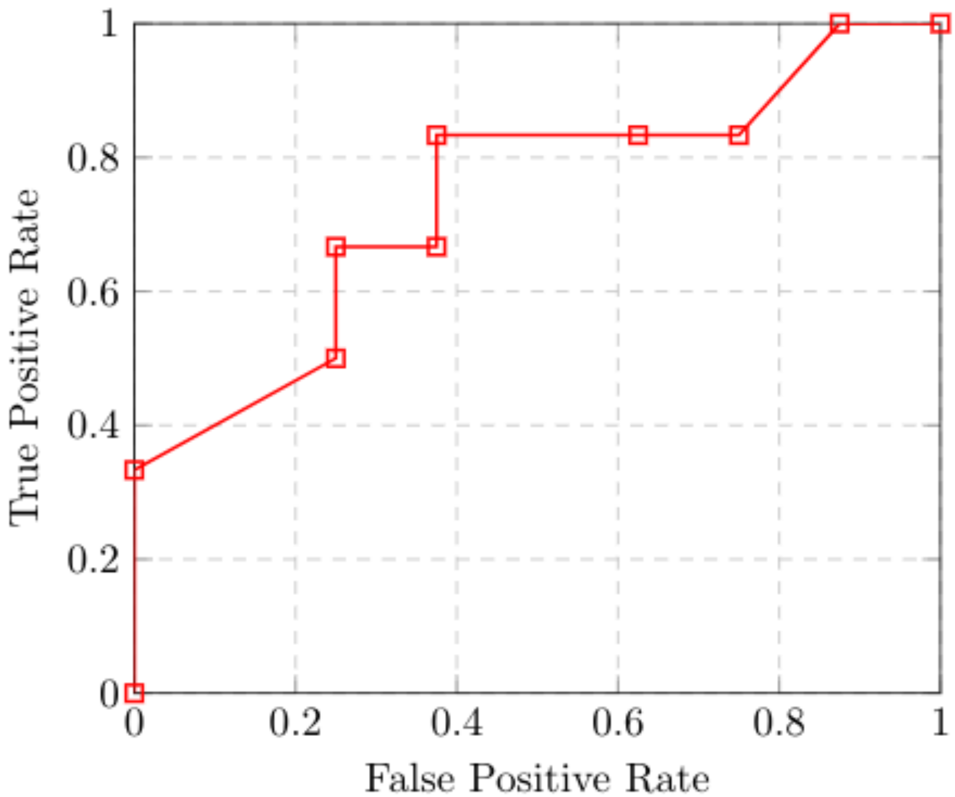
Solution

attack packet lengths: 1, 1, 2, 3, 5, 8

non-attack packet lengths: 2, 2, 4, 6, 6, 7, 8, 9

$$TP\% = TPR = \frac{TP}{TP + FN}$$

$$FP\% = FPR = \frac{FP}{FP + TN}$$



T	0	1	2	3	4	5	6	7	8	9
TP	0	2	3	4	4	5	5	5	6	6
TP%	0.00	33.33	50.00	66.67	66.67	83.33	83.33	83.33	100.00	100.00
FP	0	0	2	2	3	3	5	6	7	8
FP%	0.00	0.00	25.00	25.00	37.50	37.50	62.50	75.00	87.50	100.00

Problems with IDSes

- VERY difficult to get both good recall and precision
- Malware comes in small packages
- Looking for one packet in a million (billion? trillion?)
- If insufficiently sensitive, IDS will miss this packet (low recall)
- If overly sensitive, too many alerts will be raised (low precision)

Snort

- Open source IDS
- Signature detection
- Lots of available rulesets
  - alert tcp $EXTERNAL_NET any -> $SQL_SERVERS 3306 (msg:"MYSQL root login attempt"; flow:to_server,established; content:"|0A 00 00 01 85 04 00 00 80|root|00|"; classtype:protocol-command-decode; sid:1775; rev:2;)
  - alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-PHP Setup.php access"; flow:to_server,established; uricontent:"/Setup.php"; nocase; reference:bugtraq,9057; classtype:web-application-activity; sid:2281; rev:2;)

Defenses thus far

- **Firewalls** and **Intrusion Prevention Systems** prevent malicious packets from entering the network (in theory)
- **Intrusion Detection Systems** alert network administrators to intrusion attempts
- Both systems work best when malware is well-understood and easily fingerprinted

How do we learn about and study malware?

Honeypots

- **Honeypot:** a controlled environment constructed to trick malware into thinking it is running in an unprotected system
  - collection of decoy services (fake mail, web, ftp, etc.)
  - decoys often mimic behavior of unpatched and vulnerable services

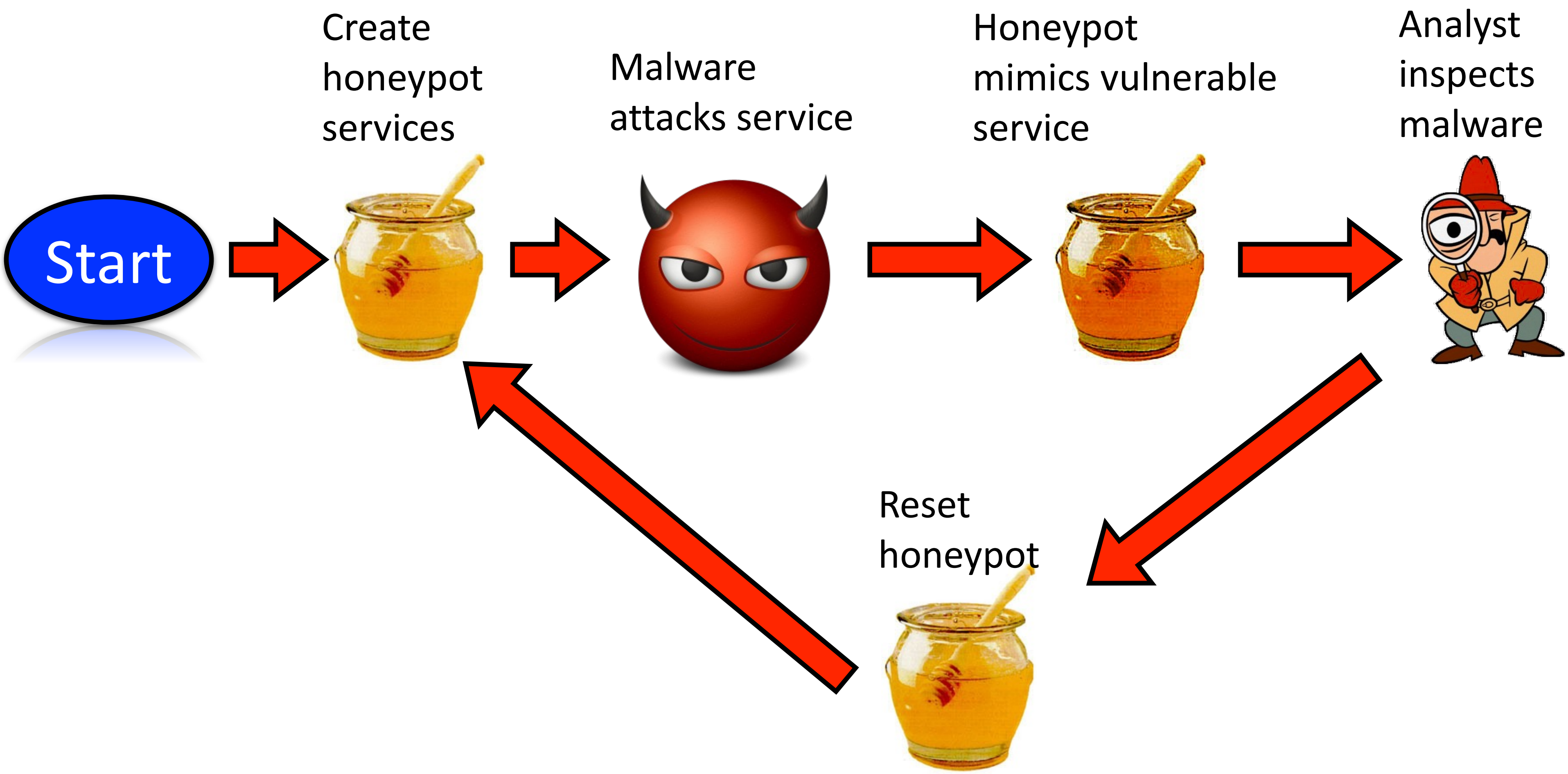
Honeypots

- Three main uses:
  - forensic analysis: better understand how malware works; collect evidence for future legal proceedings
  - risk mitigation:
    - provide "low-hanging fruit" to distract attacker while safeguarding the actually important services
    - tarpits: provide very slow service to slow down the attacker
  - malware detection: examine behavior of incoming request in order to classify it as benign or malicious

Honeypots

- Two main types:
  - **Low-interaction:** emulated services
    - inexpensive
    - may be easier to detect
  - **High-interaction:** no emulation; honeypot maintained inside of real OS
    - expensive
    - good realism

Example Honeypot Workflow



Examining Malware

- **Trace system calls:**
  - most OSes support method to trace sequence of system calls
    - e.g., ptrace, strace, etc.
  - all "interesting" behavior (e.g., networking, file I/O, etc.) must go through system calls
  - capturing sequence of system calls (plus their arguments) reveals useful info about malware's behavior

Tracing System Calls

```
% strace ls
open("/proc/filesystems", O_RDONLY)     = 3
fstat(3, {st_mode=S_IFREG|0444, st_size=0, ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f88345a4000
read(3, "nodev\tsysfs\nnodev\trootfs\nnodev\tb"..., 1024) = 346
read(3, "", 1024)                       = 0
close(3)                                = 0
munmap(0x7f88345a4000, 4096)            = 0
open("/usr/lib/locale/locale-archive", O_RDONLY) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=2772576, ...}) = 0
mmap(NULL, 2772576, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f88330f9000
close(3)                                = 0
ioctl(1, SNDCTL_TMR_TIMEBASE or TCGETS, {B38400 opost isig icanon echo ...}) = 0
ioctl(1, TIOCGWINSZ, {ws_row=24, ws_col=80, ws_xpixel=0, ws_ypixel=0}) = 0
open(".", O_RDONLY|O_NONBLOCK|O_DIRECTORY|O_CLOEXEC) = 3
fcntl(3, F_GETFD)                       = 0x1 (flags FD_CLOEXEC)
getdents(3, /* 36 entries */, 32768)    = 1104
getdents(3, /* 0 entries */, 32768)     = 0
close(3)                                = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f88345a4000
write(1, "mail  R  shared  tmp  work\n", 27) = 27
close(1)                                = 0
munmap(0x7f88345a4000, 4096)            = 0
close(2)                                = 0
exit_group(0)                           = ?
```

Examining Malware

- **Observe filesystem changes and network IO:**
  - “diff” the filesystem before and after
    - which files are the malware reading/writing?
  - capture network packets
    - to whom is the malware communicating

Examining Malware

- **Utilize hidden kernel module:**
  - capture all activity
  - challenge: encryption

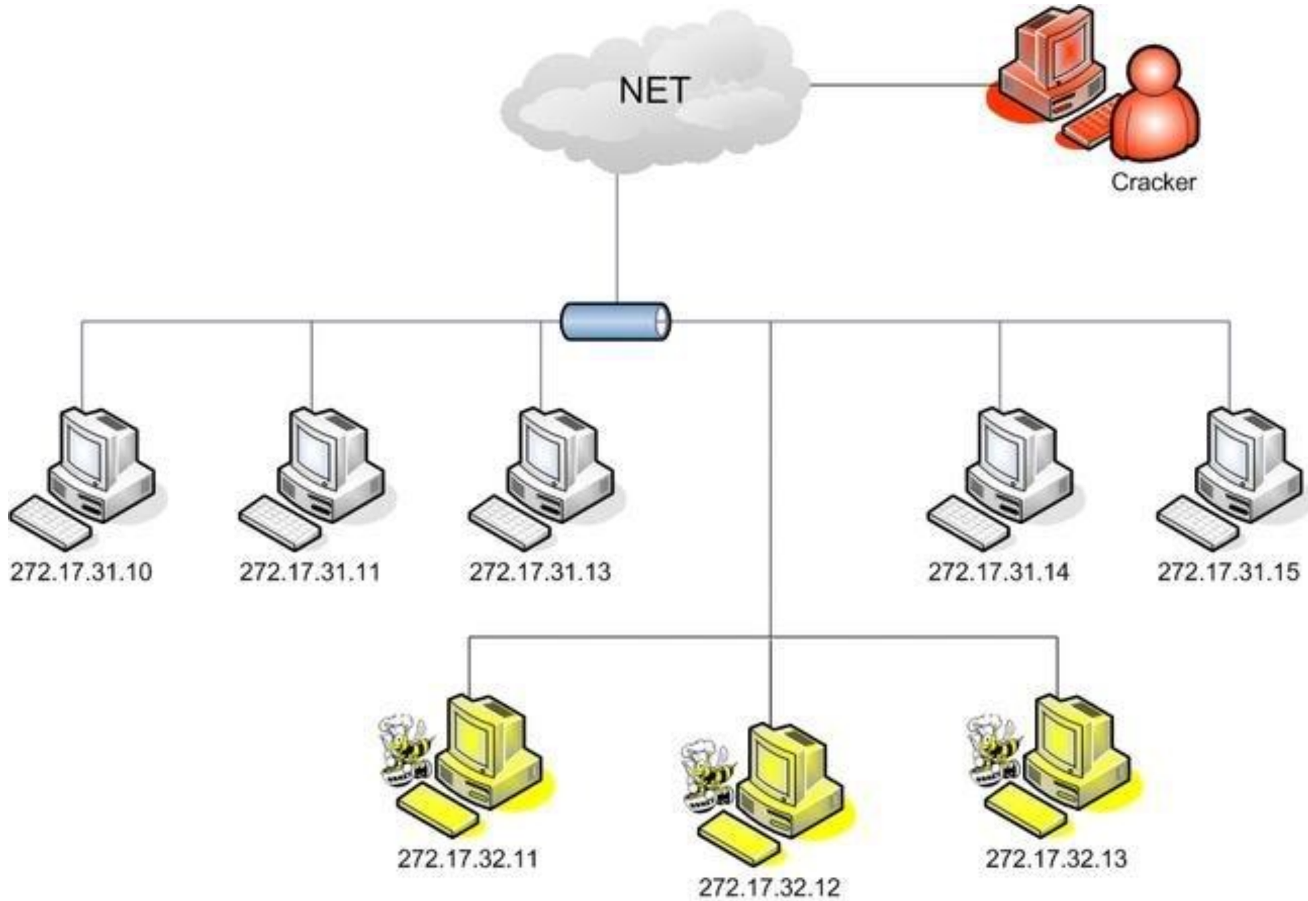
Challenges

- Honeypot *must resemble actual machine*
  - simulate actual services (Apache, MySQL, etc.)
  - but not too much... bad form to actually help propagate the worm (legal risks!)
- Some worms do a reasonably good job of detecting honeypots

Honeynets

- **Honeynet:** also called **honeyfarms**
  - Collection of honeypots that simulate a network; or
  - Single honeypot that emulates services on multiple emulated "machines" (that is, on a network)

Example Deployment



honeyd

- Open-source virtual honeynet
  - creates virtual hosts on network
  - services actually run on a single host
  - scriptable services

honeyd example:
FTP service (ftp.sh)

```
echo "$DATE: FTP started from $1 Port $2" >> $log
echo -e "220 $host.$domain FTP server (Version wu-2.6.0(5) $DATE) ready."
...
case $incmd_nocase in

        QUIT* )
           echo -e "221 Goodbye.₩r"
           exit 0;;
        SYST* )
           echo -e "215 UNIX Type: L8₩r"
           ;;
        HELP* )
           echo -e "214-The following commands are recognized (* =>'s unimplemented).₩r"
           echo -e "   USER    PORT    STOR    MSAM*   RNTO    NLST    MKD     CDUP₩r"
           echo -e "   PASS    PASV    APPE    MRSQ*   ABOR    SITE    XMKD    XCUP₩r"
           echo -e "   ACCT*   TYPE    MLFL*   MRCP*   DELE    SYST    RMD     STOU₩r"
           echo -e "   SMNT*   STRU    MAIL*   ALLO    CWD     STAT    XRMD    SIZE₩r"
           echo -e "   REIN*   MODE    MSND*   REST    XCWD    HELP    PWD     MDTM₩r"
           echo -e "   QUIT    RETR    MSOM*   RNFR    LIST    NOOP    XPWD₩r"
           echo -e "214 Direct comments to ftp@$domain.₩r"
           ;;
```

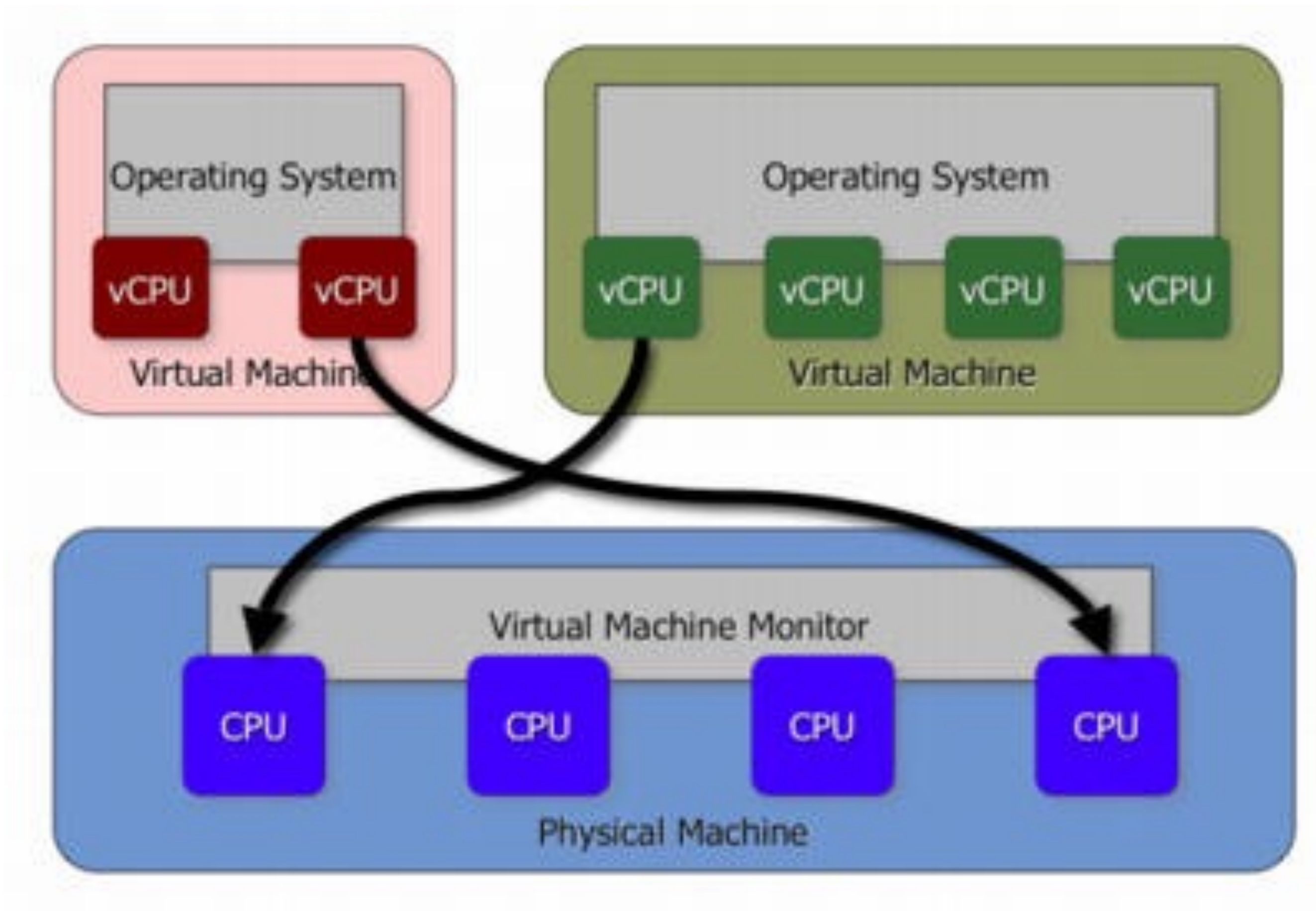
Internet Background Radiation

- **Internet Background Radiation** or **Backscatter:** Traffic that is sent to addresses on which no device is set up (these unused portions of the Internet are called **darknets**)
  - Backscatter primarily originates from spam, worms, and port scans
  - Estimated at 5.5Gbps
  - Estimated that 70% of background radiation due to Conficker Worm

Virtual Machines

- **Virtual machine:** isolated virtual hardware running within a single operating system
  - i.e., a software implementation of hardware
  - usually provides emulated hardware which runs OS and other applications
  - i.e., a computer inside of a computer
- What's the point?
  - extreme software isolation -- programs can't easily interfere with one another if they run on separate machines
  - much better hardware utilization than with separate machines
  - power savings
  - easy migration -- no downtime for hardware repairs/improvements

Virtual Machines



Honeypots and Virtual Machines

- Most virtual machines provide checkpointing features
  - **Checkpoint** (also called **snapshot**) consists of all VM state (disk, memory, etc.)
  - In normal VM usage, user periodically creates snapshots before making major changes
  - Rolling back ("restoring") to snapshot is fairly inexpensive
- **Checkpointing features are very useful for honeypots**
  - Let malware do its damage
  - Pause VM and safely inspect damage from virtual machine monitor
  - To reset state, simply restore back to the checkpoint

Honeypots and Virtual Machines

- Virtual Machines are also very useful for analyzing malware:
  - execute malware one instruction at a time
  - pause malware
  - easily detect effects of malware by looking at “diffs” between current state and last snapshot
  - execute malware on one VM and uninfected software on another; compare state

Detecting VMs

- Lots of research into detecting when you're in a virtual machine (i.e., *to prevent dynamic analysis*)
  - examine hardware drivers
  - time certain operations
  - look at ISA support
- Malware does this too!
  - if not in VM, wreak havoc
  - if in VM, self-destruct
- So, to be malware-free, *why not run your host in a virtualized environment?*