CS165 – Computer Security

Malware

November 12, 2025

Malware

- Adversaries aim to get code running on your computer that performs tasks of their choosing
  - This code is often called malware
- Three main challenges for adversaries
  - How do they get their malware onto your computer?
  - How do they get their malware to run?
  - How do they keep it from being detected?
- Focusing on what happens after initial exploitation

Viruses

- Is an attack that modifies programs on your host
- Approach
  - 1. Download a malware program ...
  - 2. Run the malware ...
  - 3. Searches for binaries and other code (firmware, boot sector) that it can modify ...
  - 4. Modifies these programs by adding code that the program will run

- What can an adversary do with this ability?

Viruses

- How does it work?
  - Modify executable files on your host
    - How does it do that meaningfully?

Viruses

- How does it work?
  - Modify executable files on your host
    - By knowing the executable file format
- Format for an executable file
  - Program loaders expect all binary files to comply with an executable format standard (e.g., Executable and Linkable Formation, ELF) to load a program correctly
- There are several aspects, but two are important
  - Entrypoint: location to start running your program
  - Sections: divisions of executable with code or data

Viruses

- How does it work?
  - Modify executable files on your host
    - By knowing the executable file format
- What types of modifications?
  - Overwrite the program “entrypoint”
    - Add code anywhere (e.g., new section) and change “entrypoint” to start there
  - Add a new section header and section
    - Change entry to that section to invoke
- All these were well known by the 1990s

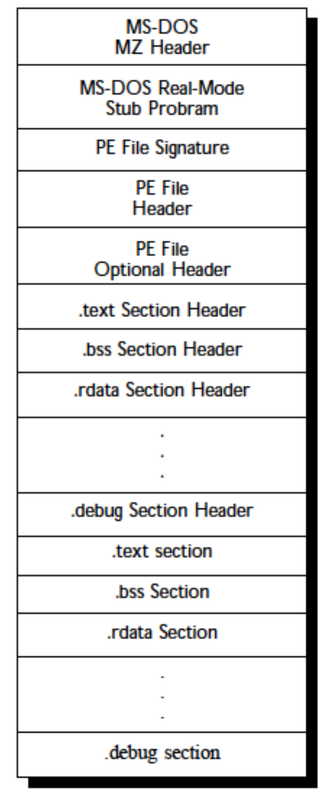


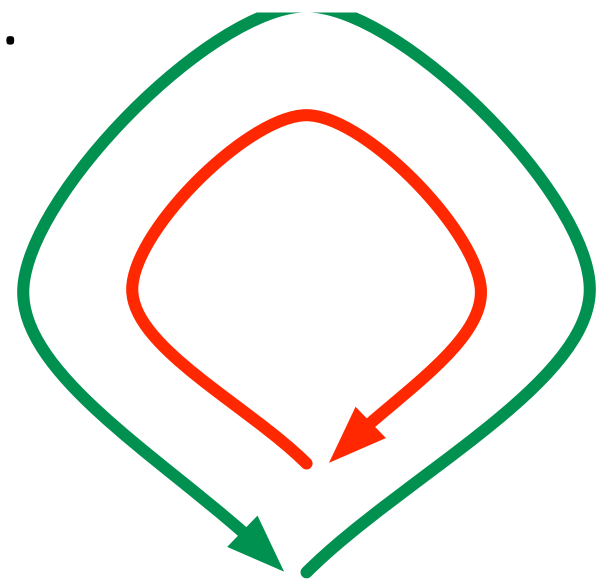
Figure 1. Overall structure of a Portable Executable file image

Virus Infection

- Keeping with the virus analogy, getting a virus to run on a computer system is called infecting the system
  - How can an adversary infect another's computer?
    - Tricking users into downloading their malware
      - E.g., Trojan horse
  - Need to also trick the user into running the malware
    - Exploiting a vulnerable program to inject code
      - E.g., memory errors
- Some systems allow an adversary to do both at once
  - E.g., phishing and email attachments

Worms

- A worm is a self-propagating program.
- As relevant to this discussion
  - 1. Exploits some vulnerability on a target host ...
  - 2. (often) embeds itself into a host ...
  - 3. Searches for other vulnerable hosts ...
  - 4. Goto (1)



- Q: Why do we care?

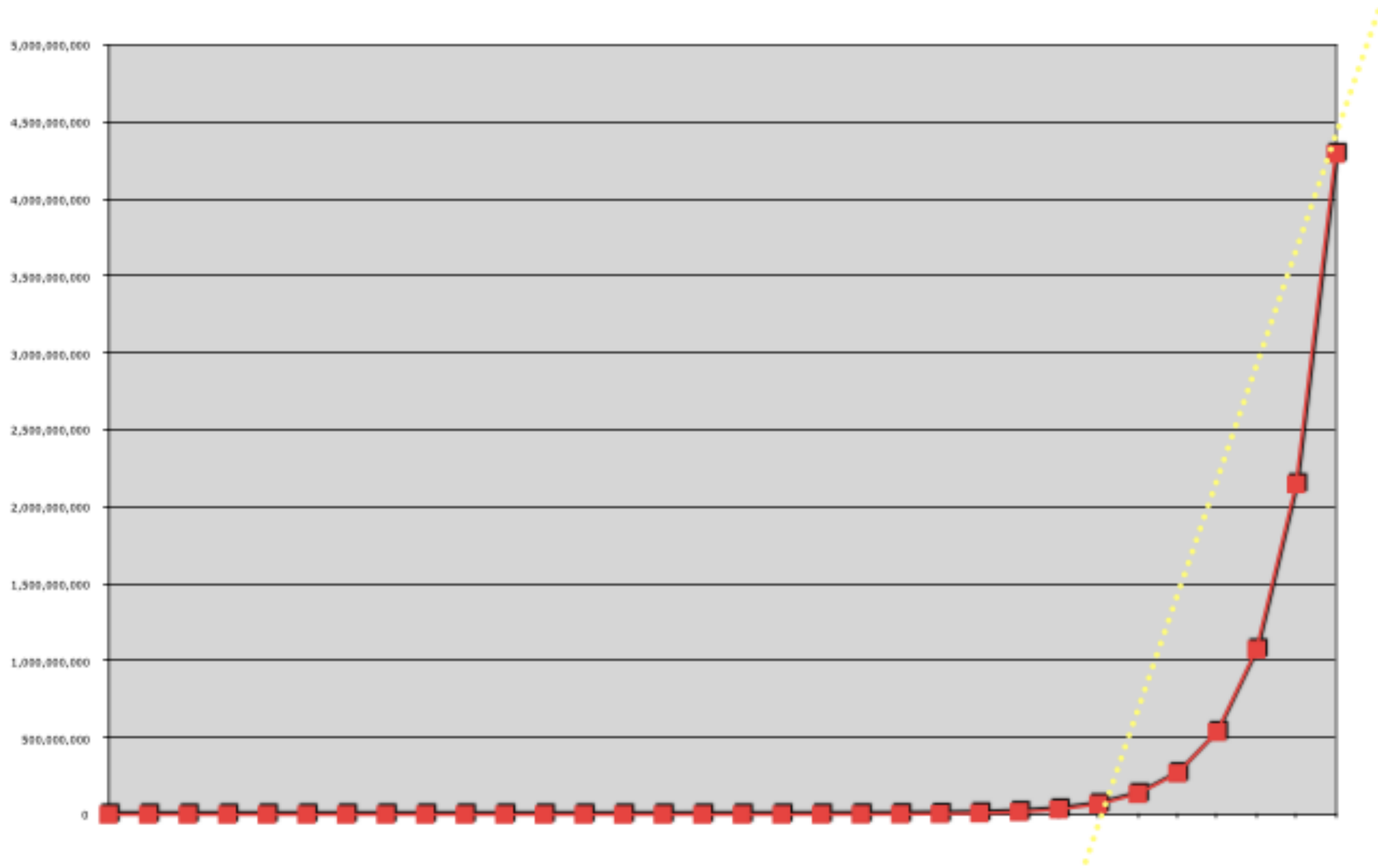
The Danger

- What makes worms so dangerous is that infection grows at an exponential rate
  - A simple model:
    - s (search) is the time it takes to find a vulnerable host
    - i (infect) is the time it takes to infect a host
  - Assume that t=0 is the worm outbreak, the number of hosts infected at t=j is?

The Danger

- What makes worms so dangerous is that infection grows at an exponential rate
  - A simple model:
    - s (search) is the time it takes to find vulnerable host
    - i (infect) is the time it takes to infect a host
  - Assume that t=0 is the worm outbreak, the number of hosts infected at t=j is
    - $2^{j/(s+i)}$
- For example, if (s+i = 1), how many infected hosts at time j=32?

The Result

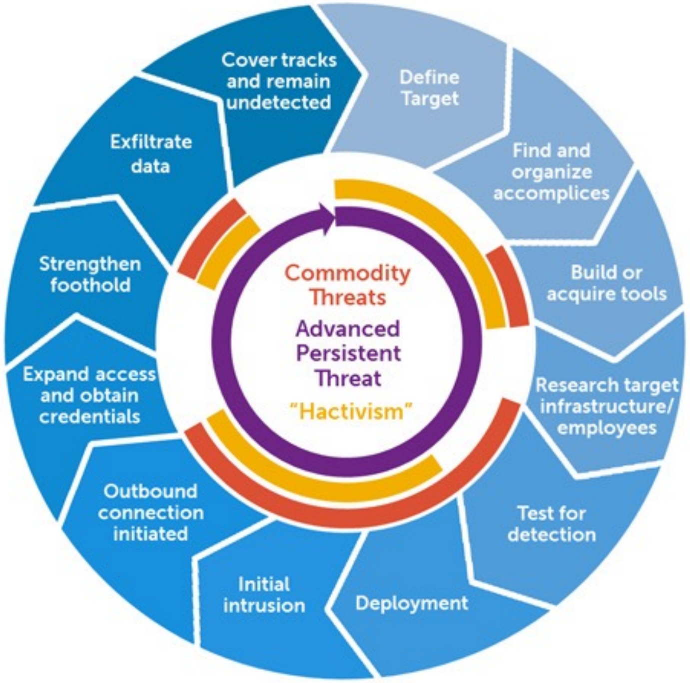

5,000,000,000
4,500,000,000
4,000,000,000
3,500,000,000
3,000,000,000
2,500,000,000
2,000,000,000
1,500,000,000
1,000,000,000
500,000,000
0

Worm Impact

- In the early days, an attacker could exploit a single vulnerability to compromise many machines
  - E.g., Code Red
- Today, worm capabilities are adapted more stealthily

Modern Malware

- Now, malware has a much greater level of sophistication
  - Now we speak of ...
  - Advanced Persistent Malware



Example: Sirefef

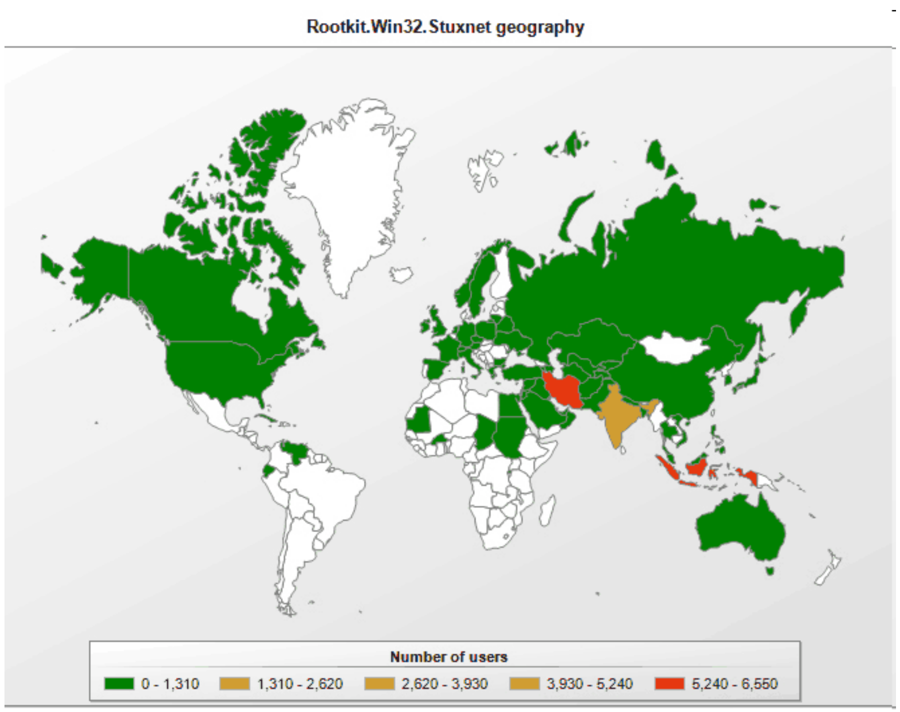
- Windows malware – from fake software update
- Technical summary
  - https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Virus:Win32/Sirefef.R
  - Attack:"Sirefef gives attackers full access to your system"
  - Runs as a Trojan software update (GoogleUpdate)
  - Runs on each boot by setting a Windows registry entry
- Does a variety of malicious things
  - Downloads code to run C&C communication
  - Some versions replace device drivers
  - Steal software keys and crack password for software piracy
  - Downloads other files to propagate the attack to other computers

Example: Sirefef

- Stealthy: “while using stealth techniques in order to hide its presence”
  - “altering the internal processes of an operating system so that your antivirus and anti-spyware can't detect it.”
  - Disables defenses, such as: Windows firewall, Windows defender
  - Changes: Browser settings
  - Changes: Windows registry
    - Resets registry change if manually “fixed”
- Microsoft: “This list is incomplete”

Example: Stuxnet

- Slides from Symantec



Example: Stuxnet

Stuxnet: Overview

- June 2010: A worm targeting Siemens WinCC industrial control system.
- Targets high speed variable-frequency programmable logic motor controllers from just two vendors: Vacon (Finland) and Fararo Paya (Iran)
- Only when the controllers are running at 807Hz to 1210Hz. Makes the frequency of those controllers vary from 1410Hz to 2Hz to 1064Hz.
- http://en.wikipedia.org/wiki/Stuxnet

Example: Stuxnet

- Very carefully designed malware for a specific industrial control environment
  - Fake update using stolen keys from a Windows driver vendor
  - Compromise/disable a variety of antivirus software to evade detection
  - Self-spreading through USB drives installed on infected computers to propagate in an air-gapped system
  - Infect application used to program the programmable logic controllers of centrifuges to inject malicious code
  - Erase malicious code from application's code viewer

Example: Stuxnet

- Stuxnet includes several modern malware facets
  - Reconnaissance: Learn the victim configuration
  - Initial Action (Infection): Trojan device driver and PLC programming application
  - Defense Evasion (Stealth): Knock out antivirus detection and remove malicious code from GUI
  - Propagation (worm): Through USB drives – no network
- Called a "kill chain" – see MITRE ATT&CK (https://attack.mitre.org)
  - Lesson: Well-funded adversaries can be difficult to stop

Intrusion Detection

- Industry has developed techniques to malware when installed on your system
- Called intrusion detection systems
  - Detect malware and evidence of compromise indicative of malware or hijacked process
- Intrusion detection has become a big business, but the problem is a significant challenge

Intrusion Detection Systems

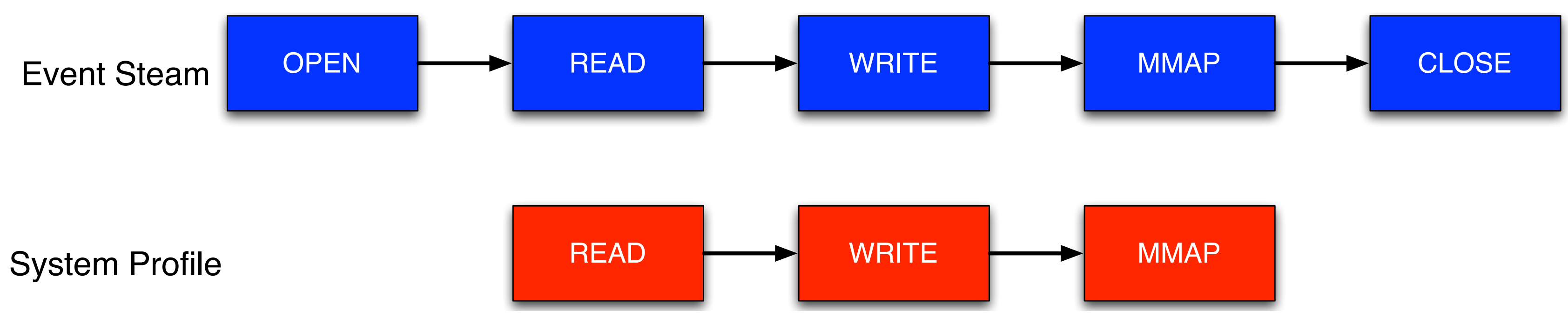
- An intrusion detection system (IDS) finds intrusions
  - "The IDS approach to security is based on the assumption that a system will not be secure, but that violations of security policy (intrusions) can be detected by monitoring and analyzing system behavior." [Forrest 98]
- However you do it, it requires
  - Training the IDS (training)
  - Looking for intrusions (detection)
- This remains an active area of computer security, that has led to an entire industry

Anomaly Detection

- Anomaly detection is one approach in IDSs
  - Compares profile of normal systems operation to monitored state
  - Hypothesis: any attack causes enough deviation from the normal operation profile (generally true?)
- Q: How do you derive normal operation?
  - Expert: construct profile from domain knowledge
  - AI: learn operational behavior from training data
  - Runtime: run the programs (a lot)
- Pitfall: abnormal behavior may not be an attack

System Call Anomaly Detection

- Idea: match sequence of syscalls made by each program with normal profiles
  - n-grams of system call sequences (learned from normal)
    - Use n-grams of length 5, 6, 11
  - Match sliding windows of sequences
  - If found, then it is normal (w.r.t. learned sequences)
  - Otherwise, assumed to be an attack (true?)

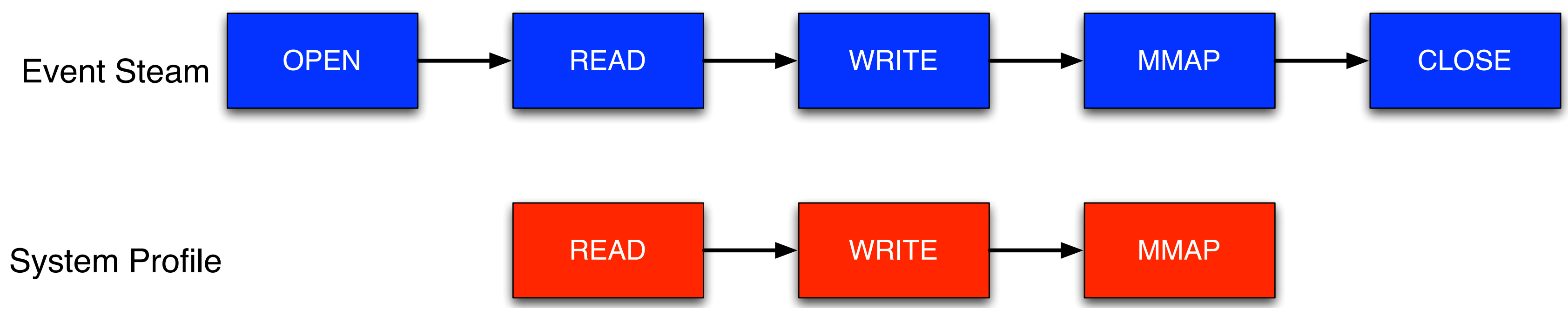


Misuse Detection

- Misuse detection is another approach in IDSs
- Monitor the operation for known attack behaviors
  - Hypothesis: attacks of the same kind has enough similarity to distinguish from normal behavior
  - This is largely pattern matching
- Q: Where do “known attack patterns” come from?
  - Record: examples of known attacks
  - Expert: domain knowledge
  - AI: Learn by negative and positive feedback
- Pitfall: May miss new attack types

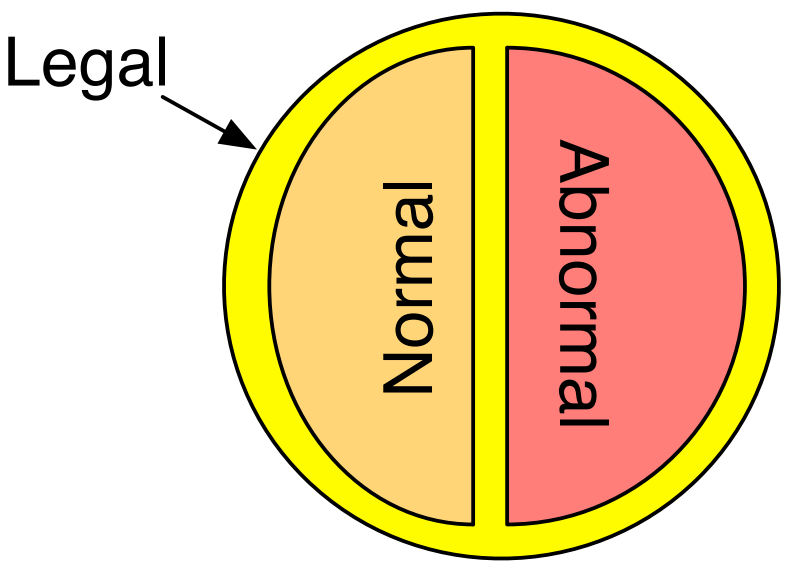
System Call Misuse Detection

- Idea: match sequence of syscalls of a program with attack profiles
  - n-grams of system call sequences (learned from attacks)
    - Use n-grams of length 5, 6, 11
  - Match sliding windows of sequences
  - If found, detected as an attack (w.r.t. learned sequences)
  - Otherwise, then assume it is normal (true?)



The “Confusion Matrix”

- What constitutes an intrusion is really just a matter of definition
  - A system can exhibit all sorts of behavior



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

Quality determined by consistency with a given definition
– which is context sensitive

“Gedanken Experiment”

- Assume a very good anomaly detector (99%)
  - And a pretty constant attack rate, where you can observe 1 out of 10,000 events are malicious

Bayes' Rule

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

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

The Base-Rate Bayesian Fallacy

- Setup
  - Pr(T) is attack probability, 1/10,000 or Pr(T) = .0001
  - Pr(F) is probability of event flagging, unknown
  - Pr(F|T) is 99% accurate (higher than most techniques)
  - Pr(F|T) = .99, Pr(!F|T) = .01, Pr(F|!T) = .01, Pr(!F|!T) = .99
- Goal: Deriving Pr(F)
  - Pr(F) = Pr(F|T)*Pr(T) + Pr(F|!T)*Pr(!T)
  - Pr(F) = (.99)(.0001) + (.01)(.9999) = .010098
- Now, what's Pr(T|F)?

The Base-Rate Bayesian Fallacy

- Now plug it in to Bayes Rule

$$\Pr(T|F) = \frac{\Pr(F|T)\ \Pr(T)}{\Pr(F)} = \frac{\Pr(.99)\Pr(.0001)}{\Pr(.010098)} = .0098$$

- So, a 99% accurate detector leads to ...
  - 1% accurate detection.
  - With 99 false positives per true positive
- This is a central problem with IDS
  - Suppression of false positives real issue
  - Open question that makes some IDSs unusable

When Is Anomaly Detection Useful?

System	Attack Density P(T)	Detector Flagging Pr(F)	Detector Accuracy Pr(F|T)	True Positives P(T|F)
A	0.1		0.65	
B	0.001		0.99	
C	0.1		0.99	
D	0.00001		0.99999	

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

When Is Anomaly Detection Useful?

System	Attack Density P(T)	Detector Flagging Pr(F)	Detector Accuracy Pr(F|T)	True Positives P(T|F)
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

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

Conclusions

- Adversaries ultimately aim to run their code (malware) on victim systems
- In the early days, infection (viruses) and propagation (worms) were relatively straightforward
- And aims to remain undetected (stealthy) and stay resident on the victim system (persistent)
  - Advanced persistent threats
- Intrusion detection aims to detect malware and compromised processes (challenging task)

Questions