https://brown-csci1660.github.io

CS1660: Intro to Computer Systems Security
Spring 2026

Lecture 19: Network Security I

Instructor: **Nikos Triandopoulos**

April 9, 2026

BROWN

CS1660: Announcements

- Course updates
  - Completed: Project 1, project 2, **project 3**, HW1, HW2, midterm
  - To be completed
    - Project 4, HW3, HW4, final
    - 3 + 1 weeks left
      - April 7, 9: Software Security + Network Security
      - April 14, 16: Network Security
      - **April 21, 23: Demos + Special Topics**
      - **April 28: Revision**
      - **April 30: Final exam**

Last class

- Cryptography
- Authentication
- Web security
- OS security
  - Access control, OS access control, file-system access control
  - Software security: More on race conditions, isolation, malware
  - Cloud security
- Network security
  - Background

Today

- Cryptography
- Authentication
- Web security
- OS security
  - Access control, OS access control, file-system access control
  - Software security: More on race conditions, isolation, malware
  - More on malware, malware detection
- Network security
  - Related topics: cloud security, email security, APTs, SIEM & security analytics
  - Background, specific attacks, …

19.1 More on malware

Recall: Malware

- Programs planted by an agent with malicious intent
  - to cause unanticipated or undesired effects
- Virus
  - a program that can replicate itself
    - pass on malicious code to other non-malicious programs by modifying them
- Worm
  - a program that spreads copies of itself through a network
- Trojan horse
  - code that, in addition to its stated effect, has a second, nonobvious, malicious effect

Recall: Some malware attack vectors

Phishing

- Attempt to fraudulently acquire sensitive information
  - Passwords, credit card numbers, etc.
- Usually copies the HTML of a website and tries to pass off as a sub-site of that page
- Phishers create a page or e-mail (spam) that appears to be from another source
- Usually relies on the user not exploring the page in depth

Recall: Inflicted harm

- Harm to users and systems
  - Sending email to user contacts
  - Deleting or encrypting files
  - Modifying system information, such as the Windows registry
  - Stealing sensitive information, such as passwords
  - Attaching to critical system files
  - Hide copies of malware in multiple complementary locations
- Harm to the world
  - Some malware has been known to infect millions of systems, growing at a geometric rate
  - Infected systems often become staging areas for new infections

Infection types

Overwriting

- Destroys original code

Pre-pending

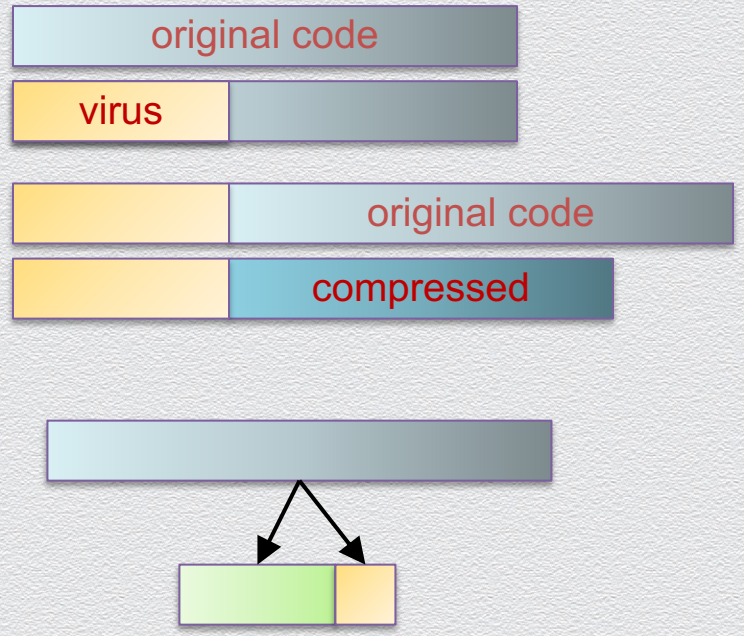
- Keeps original code, possibly compressed

Infection of libraries

- Allows virus to be memory resident
- E.g., kernel32.dll

Macro viruses

- Infects MS Office documents
- Often installs in main document template



Worm development

- Identify vulnerability still unpatched
- Write code for
  - Exploit of vulnerability
  - Generation of target list
    - Random hosts on the internet
    - Hosts on LAN
    - Divide-and-conquer
- Installation & execution of payload
- Querying & reporting on host infection
- Initial deployment on botnet

Worm template

- Generate target list
- For each host on target list
  - Check if infected
  - Check if vulnerable
  - Infect & recur
- Distributed graph search algorithm
  - Forward edges: infection
  - Back edges: already infected or not vulnerable

Concealment

Encrypted virus

- Decryption engine + encrypted body
- Randomly generate encryption key
- Detection looks for decryption engine

Polymorphic virus

- Encrypted virus with random variations of the decryption engine (e.g., padding code)
- Detection using CPU emulator

Metamorphic virus

- Different virus bodies
- Approaches include
  - Code permutation
  - Instruction replacement
- Detection is challenging

Rootkits

A rootkit modifies the operating system to hide its existence

- E.g., modifies file system exploration utilities (e.g., ls, cd, …)
- Hard to detect using software that relies on the OS itself

RootkitRevealer for Windows

- By Bryce Cogswell and Mark Russinovich (Sysinternals)
- Two scans of file system
- High-level scan using the Windows API
- Raw scan using disk access methods
- Discrepancy reveals presence of rootkit
- Could be defeated by rootkit that intercepts and modifies results of raw scan operations

Countermeasures for users

- Use software acquired from reliable sources
- Test software in an isolated environment
- Only open attachments when you know them to be safe
- Treat every website as potentially harmful
- Create and maintain backups

Countermeasures for developers

- Modular code: Each code module should be
  - Single-purpose
  - Small
  - Simple
  - Independent
- Encapsulation
- Information hiding
- Mutual suspicion
- Confinement
- Genetic diversity

Code testing

- Unit testing
- Integration testing
- Function testing
- Performance testing
- Acceptance testing
- Installation testing
- Regression testing
- Penetration testing

Design principles for security

- Least privilege
- Economy of mechanism
- Open design
- Complete mediation
- Permission based
- Separation of privilege
- Least common mechanism
- Ease of use

Other countermeasures

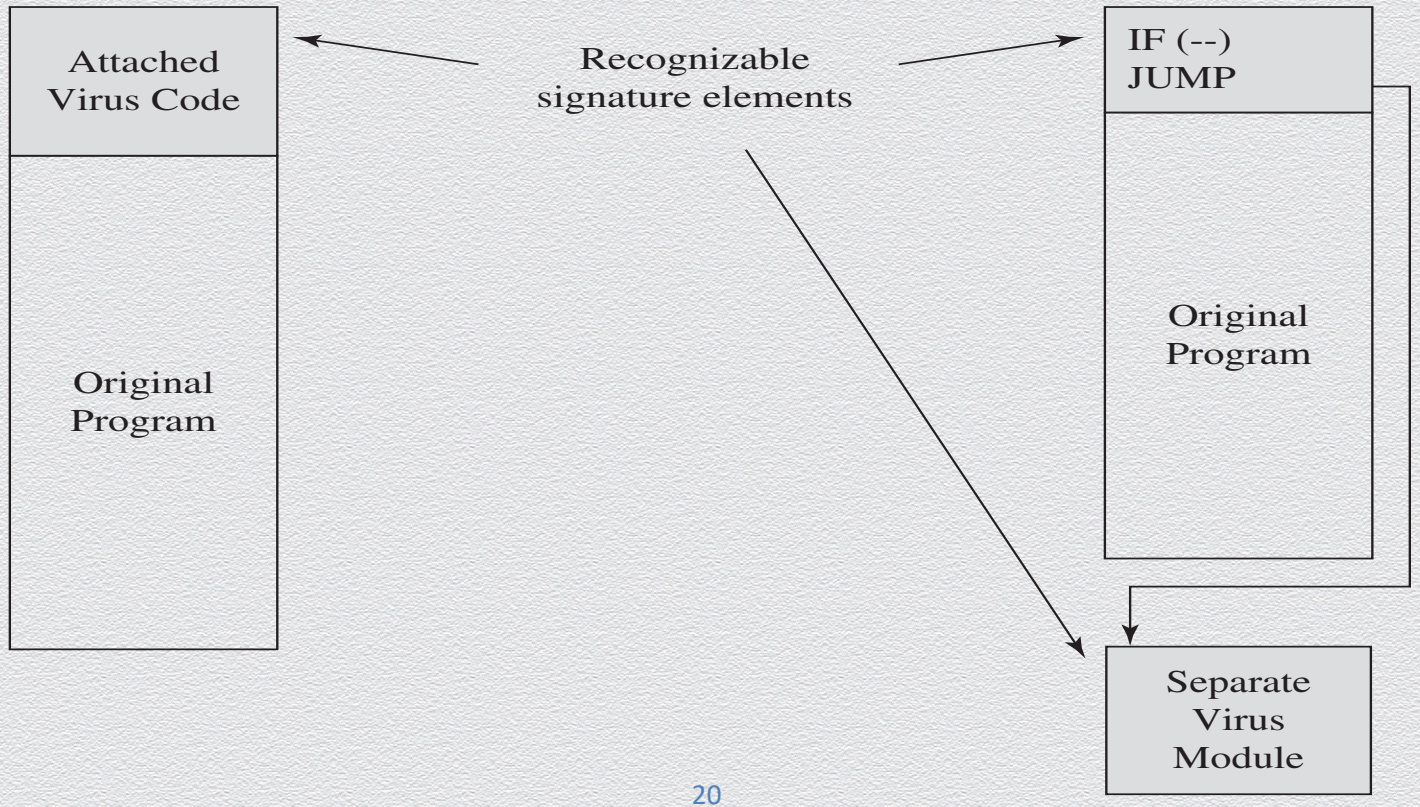
- Good
  - Proofs of program correctness—where possible
  - Defensive programming
  - Design by contract
- Bad
  - Penetrate-and-patch
  - Security by obscurity

19.2 Malware detection

Virus detection

- Virus scanners look for signs of malicious code infection using signatures in program files and memory
- Traditional virus scanners have trouble keeping up with new malware—detect about 45% of infections
- Detection mechanisms
  - Known string patterns in files or memory
  - Execution patterns
  - Storage patterns

Virus signatures



Undecidability

Undecidable problem

- A yes/no problem for which there exists no algorithm that always returns an answer

Halting Problem

- Will an arbitrary program eventually return an output (i.e., halt or terminate)?
- Alan Turing (1936) proved that this is undecidable

A large class of decision problems can be proved to be undecidable!

Why the halting problem is undecidable?

Suppose algorithm halts(P) can decide if any program P halts.

We can show by contradiction that no such algorithm exists.

def prog():

- if halts(prog) then loop_forever()

There are 2 cases:

- If halts(prog) returns True, then prog loops forever
- If halts(prog) returns False then prog terminates

Each case leads to a contradiction

- Thus, no algorithm halts can exist

Virus detection is undecidable

Theoretical result by Fred Cohen (1987)

- Virus abstractly modeled as program that eventually executes infect()
- Code for infect() may be generated at runtime
- Proof by contradiction like that of the halting problem
- Suppose program isVirus(P) determines whether program P is a virus

def prog():

- if (not isVirus(prog)) then infect()

There are 2 cases:

- If isVirus(prog) returns True, then prog does not infect
- If isVirus(prog) returns False, then prog infects

Each case leads to a contradiction

- Thus, no algorithm isVirus can exist

Virus detection is undecidable (cont.)

Alternatively:

- Suppose program isVirus(P) determines whether program P is a virus
- def prog():

```
{
foo(); //harmless code
infect()
}
```

Let's run isVirus(prog)

- If foo() can return, isVirus should return True
- If foo() never returns, then isVirus should return False
  (because infect will never execute)

Thus, isVirus must determine whether foo() can ever halt, which contradicts that the **halting problem** is undecidable.

- Thus, no algorithm isVirus can exist

Question

Which of the following statements summarizes what it means to say that virus detection is undecidable?

A. Virus detection is theoretically possible but exceedingly difficult to program

B. Assuming the existence of a virus detection program leads to a logical contradiction

C. Virus detection is a problem whose solution requires an exponential time algorithm

D. None of the above

Answer

Which of the following statements summarizes what it means to say that virus detection is undecidable?

A. Virus detection is theoretically possible but exceedingly difficult to program

B. Assuming the existence of a virus detection program leads to a logical contradiction

C. Virus detection is a problem whose solution requires an exponential time algorithm

D. None of the above

Other undecidable detection problems

Detection of a virus

- by its appearance
- by its behavior

Detection of a triggering mechanism

- by its appearance
- by its behavior

Detection of a virus detector

- by its appearance
- by its behavior

Detection of an evolution of

- a known virus
- a known triggering mechanism
- a virus detector

But detection works where prevention fails…

Detection by its appearance

- Detects specific malicious signatures
- Often uses pattern matching techniques
- Problems
  - False negative
  - Signature evasion

Detection by its behavior

- Detects anomalies on a normal system or network activity
- Often uses machine learning
- Problems
  - False positive
  - Legitimate behavior can be atypical

Here, detection is the act of noticing or discovering something

High costs of errors

- False Positives (FP) require long and expensive analysis
- False Negatives (FN) can be catastrophic
- Examples
  - **Airport Security**: FP is when ordinary items such as keys or coins get mistaken for weapons (machine goes "beep")
  - **Quality Control**: FP is when a good-quality items get rejected; FN is when a poor-quality items get accepted
  - **Presumption of innocence:** "It is better that ten guilty persons FN escape than that one innocent suffer FP”
  - **Antivirus software**: a FP is when a normal file is thought to be a virus

Signatures

- Scan compares the analyzed object with a database of signatures
- A signature is a virus fingerprint
  - E.g., a string with a sequence of instructions specific for each virus
  - E.g., a cryptographic hash value
  - But different from a digital signature
- A file is considered infected if there is a signature inside its code
  - Fast pattern matching techniques to search for signatures
- All such signatures together create the malware database that usually is proprietary

Heuristic analysis

Useful to identify new and zero-day malware

- Code analysis
  - Based on the instructions, the antivirus can determine whether the program is malicious, i.e., program contains instruction to delete system files
- Execution emulation (sandbox)
  - Run code in isolated emulation environment
  - Monitor actions that target file takes
  - If the actions are harmful, mark as virus
- Heuristic methods can trigger false alarms

Online Vs. offline anti-virus software

Online

- Detects specific malicious signatures
- Free browser plug-in
- Authentication through third party certificate (i.e., VeriSign)
- No shielding
- Software and signatures update at each scan
- Poorly configurable
- Scan needs internet connection
- Report collected by company offering service

Offline

- Paid annual subscription
- Installed on the OS
- Software distributed securely by the vendor online or a retailer
- System shielding
- Scheduled software and signatures updates
- Easily configurable
- Scan without internet connection
- Report collected locally and possibly sent to vendor

Anti-malware software today

In addition to signature-based scanning, behavior-based detection or sandboxing, anti-malware software often relies on multiple resources for current malware in the wild

- E.g., Symantec's STAR malware protection technologies rely on the following
  - **File-Based Protection**: Still of major protection role, due to innovations in static file heuristics.
  - **Network-Based Protection**: Detects when known/unknown vulnerabilities are used against a system.
  - **Behavior-Based Protection**: Looks at dynamic behavior of malicious activities rather than static characteristics.
  - **Reputation-Based Protection**: Examines file meta-data, e.g., age, origin, travels, location, etc.
  - **Remediation**: Employs set of technologies that can help clean up an infected system.

Quarantine (aka, virus chest)

A suspicious file can be isolated in a folder or database called quarantine

- E.g., if the result of the heuristic analysis is positive and you are waiting for updates of the signatures
- The suspicious file is not deleted but made harmless
  - The user can decide when to remove it or eventually restore it in case of a false positive
  - Interacting with a file in quarantine is possible only through the antivirus program
- A file in quarantine is often stored encrypted to prevent its execution
- The quarantine system architecture is typically proprietary

Static Vs. dynamic analysis

Static

- Check the code without execution
  - **Filtering**: Scan with different antivirus and check if they return same result with different name
  - **Weeding**: Remove the correct part of files as junk to better identify the virus
  - **Code analysis**: Check binary code to understand if it is an executable
  - **Disassembling**: Check if the byte code shows something unusual

Dynamic

- Check the execution of codes inside a virtual sandbox
- Monitor
  - File changes
  - Registry changes
  - Processes and threads
  - Network ports

How to check if AV software is running?

- Eicar signature
  - X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
- www.caro.org
- www.eicar.org

Anti-virus checking

Shield

- Background process
  - service/daemon
- Scans each time a file is touched
  - open, copy, execute, etc.

On-demand

- Scan on explicit user request
- Scan according to regular schedule
- Scan on a suspicious
  - File
  - Directory
  - Drive, etc.

Anti-virus evaluation

Comparative- and performance-based

- Check the number of already known viruses that are found
- Check the time to perform the scan

False alarm test

- Compute of false viruses detected

Heuristic & behaviour test

- Measure the proactive protection capabilities

Anti-viruses are ranked using both parameters

- http://www.av-comparatives.org/

Resources

- Symantec's Internet Security Threat Report
  - Published annually
- Countdown to zero day by Kim Zetter, 2014
- Art of Computer Virus Research and Defense by Peter Szor
- http://virus.wikidot.com/

Responsible disclosure

What happens if someone discovers a vulnerability in software?

- 2008: MBTA sued three MIT students to prevent them from giving a talk about vulnerabilities in the subway fare system
- 2019: researcher Jonathan Leitschuh discovered a vulnerability in Zoom, which they did not fix until he publicly disclosed it
- Today, many companies have bug bounty programs in place to encourage responsible disclosure of vulnerabilities
- Disclosure deadlines: amount of time researchers give companies to patch vulnerabilities before disclosure
- Often varies by company and by how critical the vulnerability is

19.3 Cloud security

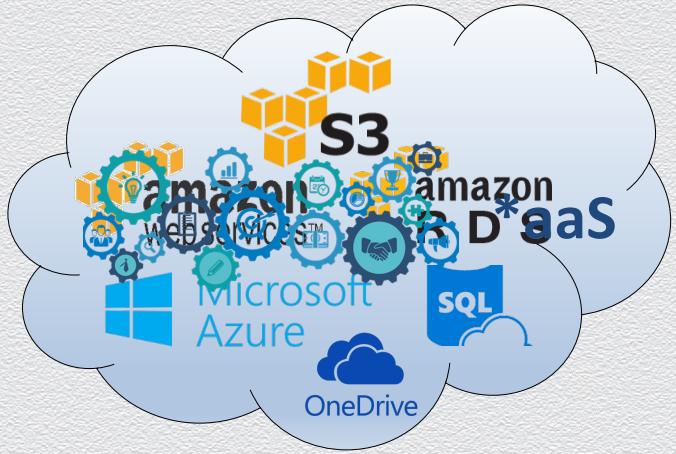
Another example: Tax return preparation…

Involves information collection & processing

- calculate financial data
  - payroll, profits, stock quotes, …
- manage data
  - search emails, store records, …
- submit – done!

… by many unknown machines!

Data & computation outsourcing



Cloud-based services

- hardware, OS, software, apps, …
- storage, computation, databases, analytics, …

Transformative multi-platform technology

- businesses, organizations or individuals
- client-server, distributed, P2P, Web-based, …

Internet protocols **social networks** **big-data analytics** **sharing economy** **FinTech**

Security consequences

Fact: Untrusted interactions

- information is processed outside one's administration control or "trust perimeter"

Risk: Falsified / leaked information

- information may (un)intentionally altered by or shared with unauthorized entities

Goal: Integrity / privacy safeguards for outsourced assets

- need to protect information against change, damage / unauthorized access

What is cloud computing?

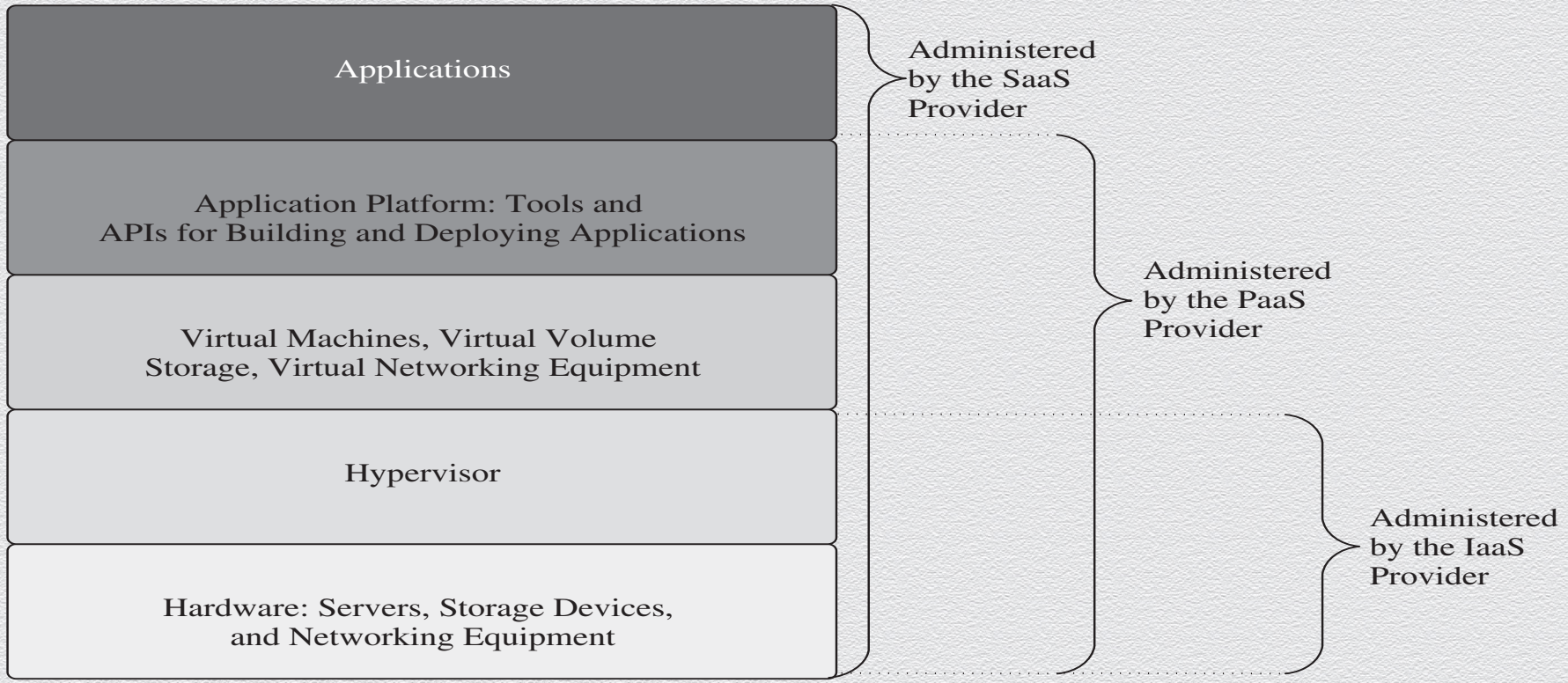
- On-demand self-service
  - Add or subtract resources as necessary
- Broad network access
  - Mobile, desktop, mainframe
- Resource pooling
  - Multiple tenants share resources that can be reassigned dynamically according to need and invisibly to the tenants
- Rapid elasticity
  - Services can quickly and automatically scale up or down to meet customer need
- Measure service
  - Like water, gas, or telephone service, usage can be monitored for billing

Cloud-computing service models

On-demand self-service computing

- Software as a service (SaaS)
  - the cloud provider gives the customer access to applications running in the cloud
- Platform as a service (PaaS)
  - the customer has his or her own applications,
    but the cloud provides the languages and tools for creating and running them
- Infrastructure as a service (IaaS)
  - the cloud provider offers processing, storage, networks, and other computing resources that enable customers to run any kind of software

Service models



Deployment models

- Private cloud
  - Infrastructure that is operated exclusively by and for the organization that owns it
- Community cloud
  - Shared by several organizations with common needs, interests, or goals
- Public cloud
  - Owned by a cloud service provider and offered to the general public
- Hybrid cloud
  - Composed of two or more types of clouds, connected by technology that enables data and applications to balance loads among those clouds

Cloud provider assessment

- Security issues to consider
  - Authentication, authorization, and access control options
  - Encryption options
  - Audit logging capabilities
  - Incident response capabilities
  - Reliability and uptime

Security benefits of cloud services

- Geographic diversity
  - many cloud providers run data centers in disparate geographic locations and mirror data across locations, providing protection from natural and other local disasters
- Platform & infrastructure diversity
  - different platforms and infrastructures mean different bugs and vulnerabilities, which makes a single attack or error less likely to bring a system down
  - using cloud services as part of a larger system can be a good way to diversify your technology stack
  - e.g., Honeywords, virtualization, etc.

Cloud-based security functions

Some security functions may be best handled by cloud service providers

- Email filtering
  - since email is already hopping through a variety of SMTP servers, adding a cloud-based email filter is as simple as adding another hop
- DDoS protection
  - cloud-based DDoS protection services update your DNS records to insert their servers as proxies in front of yours
  - they maintain sufficient bandwidth to handle the flood of attack traffic
- Network monitoring
  - cloud-based solutions can help customers deal with steep hardware requirements and can provide monitoring and incident response expertise

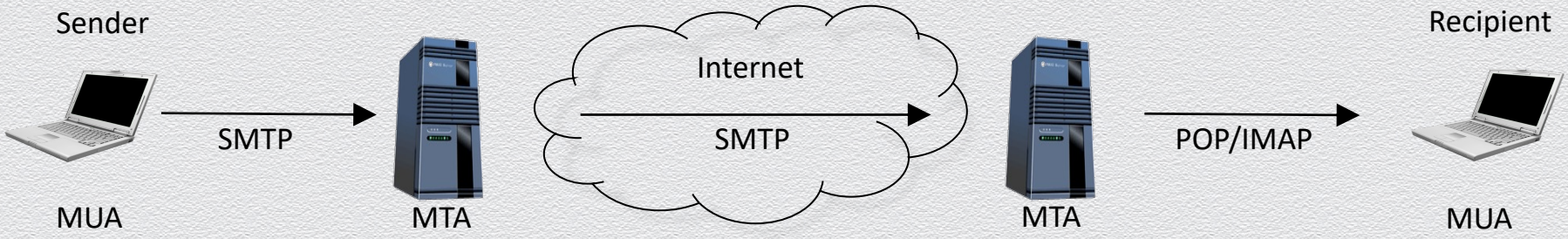
Switching cloud providers

- Switching cloud providers is expensive and difficult but sometimes becomes necessary and urgent
- It is best to have backup options in place in case a migration away from a cloud provider is necessary, but many cloud providers make that practically impossible
- SaaS providers are generally hardest to migrate away from, followed by PaaS, then IaaS

19.4 Email security

Email transport

- MUA: mail user agent, aka mail client
- MTA: mail transport agent, aka mail server



SMTP

Simple Mail Transfer Protocol

- Client connects to server on TCP port 25
- RFC 821 (1982) – 2821 (2001)
- Client sends commands to server
- Server acks or notifies of error

Security issues

- Sender not authenticated
- Message and headers transmitted in plain text
- Message and header integrity not protected
- Spoofing and spamming trivial to accomplish

Example SMTP session

HELO mail.cs.brown.edu

MAIL FROM:<POTUS@whitehouse.gov>

RCPT TO:<nikolaos_triandopoulos@brown.edu>

DATA

Subject: Executive order

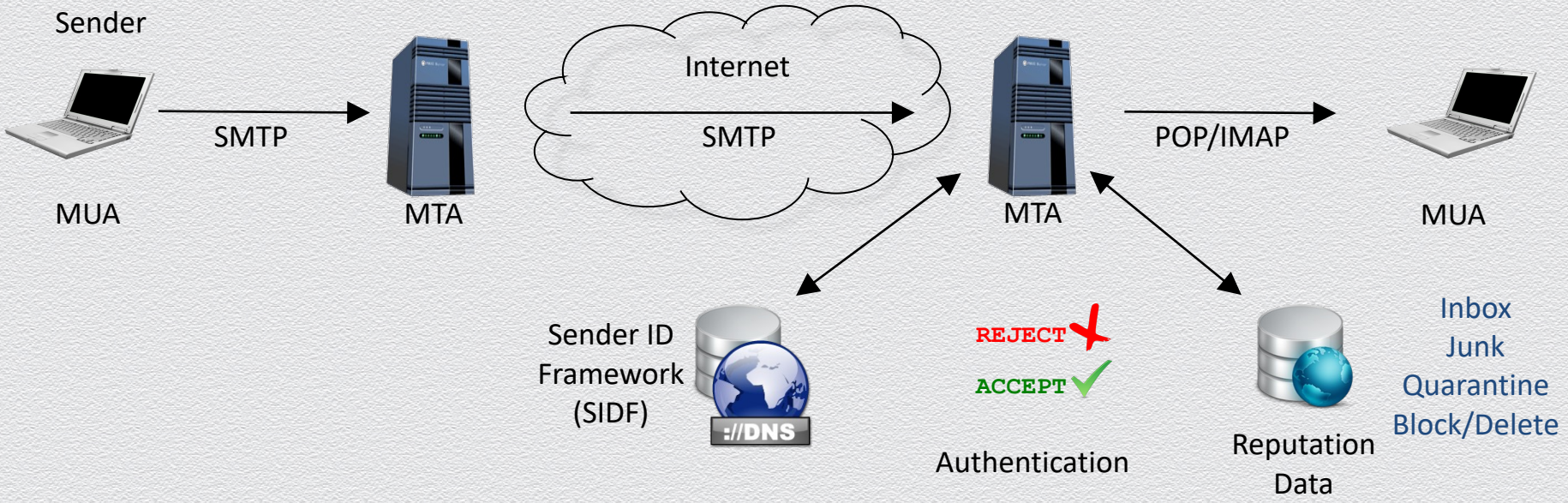
Date: Thu, April 9, 2026

You are hereby ordered to grade all the students of CS 166 class with A.

The President of the United States

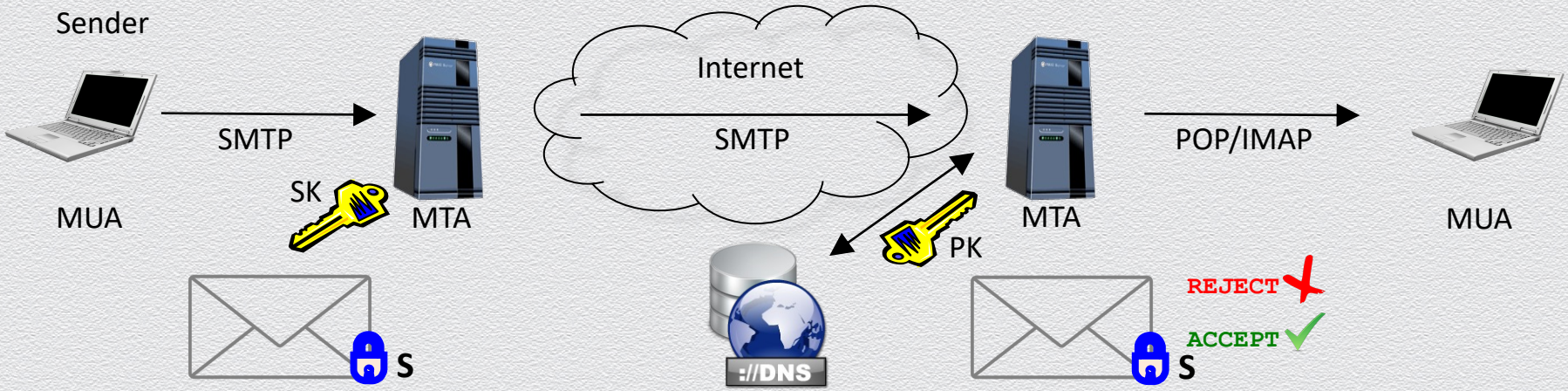
Sender ID and Sender Policy Framework (SPF)

- Store DNS records about servers authorized to send mail for a given domain
- Look up domain in From header to find IP address of authorized mail server



DomainKeys Identified Mail (DKIM)

- Sender's MTA signs email to authenticate domain; server's PK available in DNS record
- Used in conjunction with other spam filtering methods



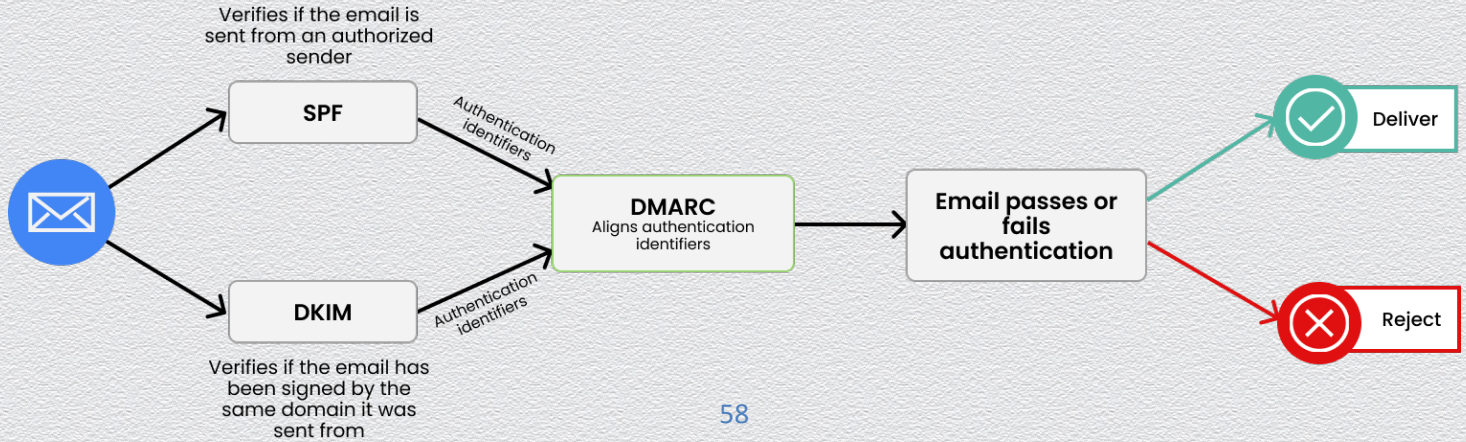
DomainKey Signature
a=rsa-sha1; s=mail; d=example.net;
c=simple; q=dns; b=Fg…5J

Authentication Result
example.net; from=bob@example.net;
domainkeys=pass;

DMARC

Domain-based Message Authentication, Reporting & Conformance

- Allows you to get reports back on the effectiveness of SPF and DKIM controls
- Validates that “From” header is the same as the domains validated by SPF and DKIM
- Provides clear instructions to the receiving server on how to handle emails that fail SPF or DKIM
- Google message header validator: https://toolbox.googleapps.com/apps/messageheader/



19.5 Advanced Persistent Threats (APTs)

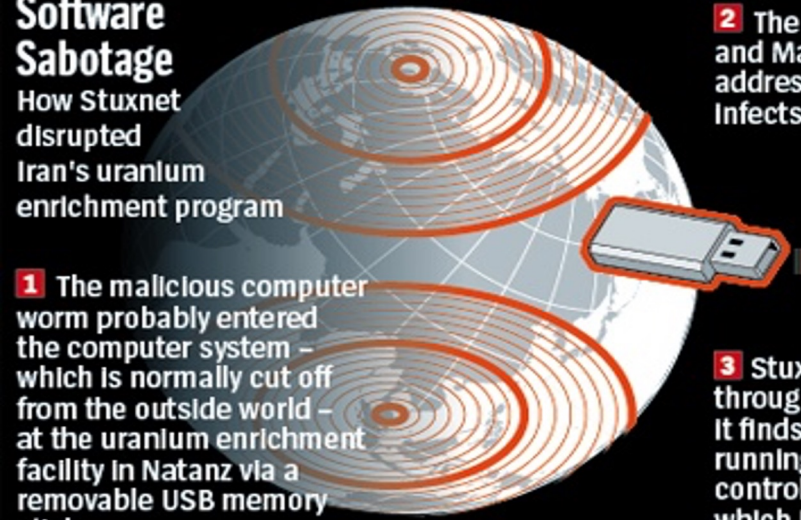
Cyberweapons

Starting from 2010 several viruses acted as a sort of weapons in international relationships

- Usually this is not confirmed by governments
- Most famous:
  - 2010 Stuxnet
  - 2012 Flame
  - 2020 Orion Solarwinds

Software Sabotage

How Stuxnet disrupted Iran's uranium enrichment program

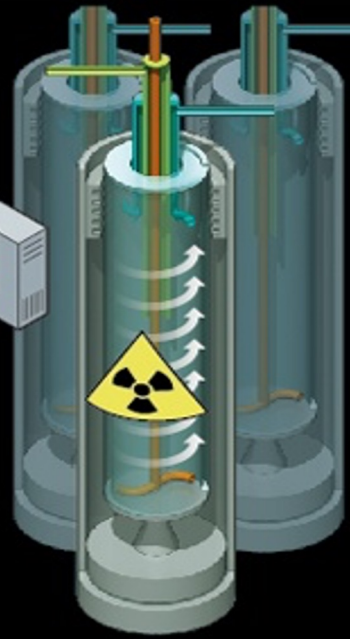


1 The malicious computer worm probably entered the computer system – which is normally cut off from the outside world – at the uranium enrichment facility in Natanz via a removable USB memory stick.

2 The virus is controlled from servers in Denmark and Malaysia with the help of two Internet addresses, both registered to false names. The virus infects some 100,000 computers around the world.

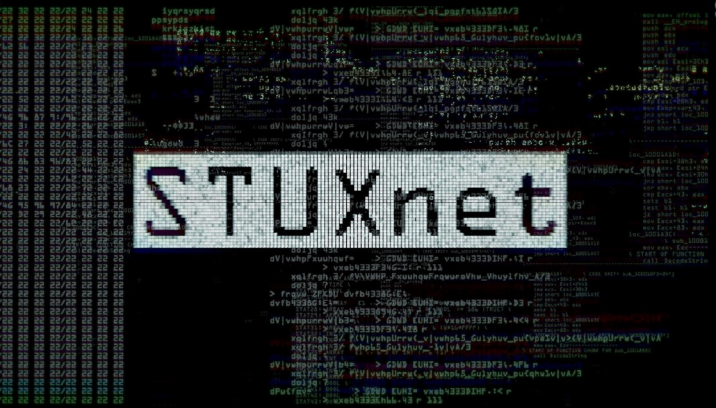
3 Stuxnet spreads through the system until it finds computers running the Siemens control software Step 7, which is responsible for regulating the rotational speed of the centrifuges.

4 The computer worm varies the rotational speed of the centrifuges. This can destroy the centrifuges and impair uranium enrichment.

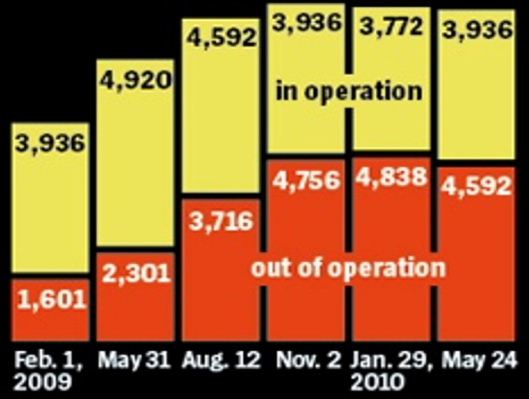


Iranian centrifuges for uranium enrichment

DER SPIEGEL



5 The Stuxnet attacks start in June 2009. From this point on, the number of inoperative centrifuges increases sharply.



Source: IAEA, ISIS, FAS, World Nuclear Association, FT research

Stuxnet: Command & Control (C&C)

Once a system was infected Stuxnet checked two fake web domains:

- mypremierfutbol.com
- todaysfutbol.com
- registered with two fake names and credit cards
- servers pointed to Denmark and Malaysia

Virus sent encrypted information about the infected target

- Windows version
- internal IP address
- targeted Siemens sw installed

If target has no Siemens sw installed

- the payload does not start
- the worm spreads to other targets

NEW "FLAME" CYBER WEAPON

Article in 2012

Kaspersky Lab, one of the world's largest makers of anti-virus software, discovered a new malware codenamed "Worm.Win32.Flame," or simply "Flame," the most complex piece of malicious software yet found.

COMPLEXITY

Comprising almost 20 MB in size and some 20 modules of code when fully deployed, Flame is one of the biggest examples of malicious software ever discovered

BREADTH

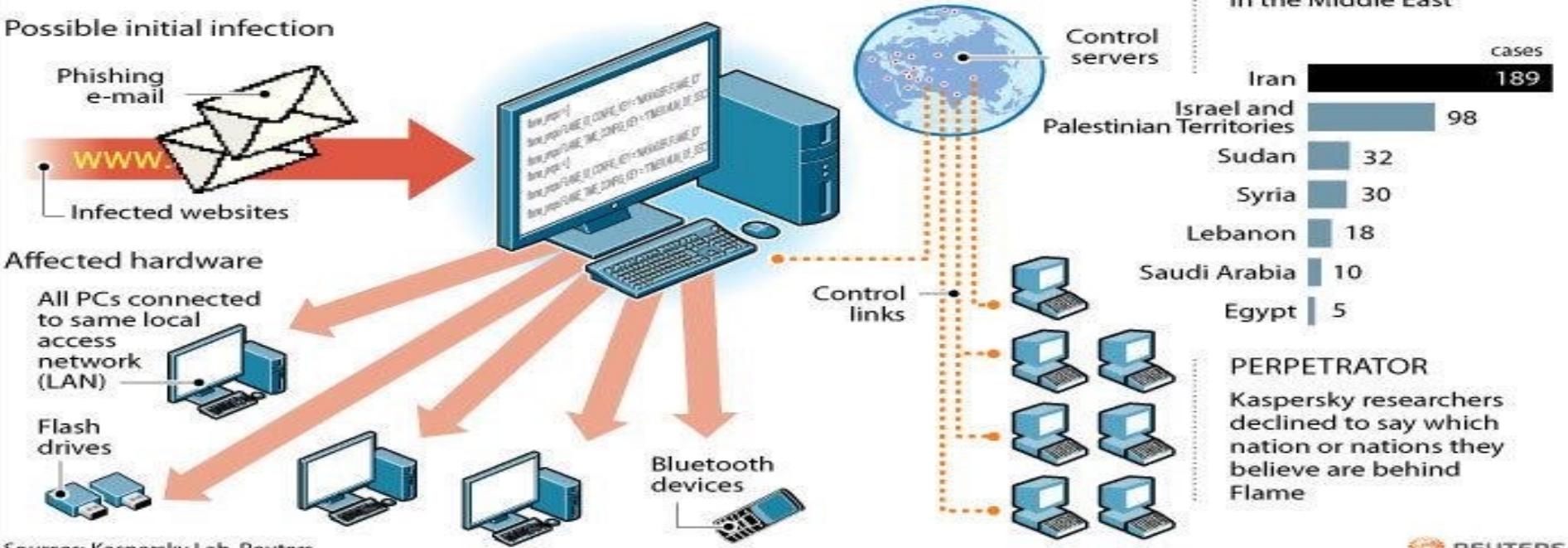
Virus can record sounds, access Bluetooth communications, capture screenshot images and log internet messaging conversations

NETWORK

The creators of the virus used a network of some 80 servers across Asia, Europe and North America to remotely control infected machines

VICTIMS

Researchers estimate that altogether between 1,000 and 5,000 machines are infected worldwide, with the larger number of infected computers found in the Middle East



PERPETRATOR

Kaspersky researchers declined to say which nation or nations they believe are behind Flame

Sources: Kaspersky Lab, Reuters

REUTERS

Flame details

- Flame used >80 domains for C&C and and innovative attack vector
  - A rogue Microsoft signed update based on a md5 hash collision
- Flame targets different office files (e.g. word, excel) and AutoCAD files
  - The malware extracted 1 KB of text from each file and transmitted it back to C&C
  - A supercomputer would likely elaborate which file could be interesting
- Patient zero?
  - Difficult to establish
  - First infection uncovered was dated Dec 2007 in Europe, but Flame could potentially alter the timestamp to prevent researchers from dating the work

The Equifax breach

Equifax is a leading credit reporting agency

- Keeps personal information and credit history for virtually every American
- In summer 2017, sensitive personal info about 148 million people was stolen
  - Name
  - Date of birth
  - Address
  - Social security number

Attackers exploited vulnerability in popular web server software

- Apache Struts code for Java web applications was vulnerable to remote code execution
- Attacker only needed a browser
- Vulnerability had existed for years
- Variants reported in March and September 2017
- First patch available in March 2017

The Equifax breach, 1 year after

- CEO resigned after the breach revelation
  - Forfeited a $3M bonus
  - Kept $18M in pension benefits
- Other executives also resigned
- No significant action taken for consumer reparation, no substantive regulatory changes since the breach
- US senators Elizabeth Warren and Mark Warner introduced a bill to hold credit agencies accountable for data theft

Stock shed about 1/3 of its value in following months, but recouped most of the loss after one year

SolarWinds hack

A supply-chain attack against an IT-management company

- Hackers able to compromise networks of many other companies and deliver malware
- Malware inserted into update of Orion, a network-management product
- Hackers used AWS as a disguise
- At least 100 companies impacted (e.g, Microsoft, US government agencies)
- FireEye, an impacted cybersecurity company discovered the hack
- SolarWinds issued a security advisory + what defensive measures could be taken
- FBI Investigation to find the actors
- “solarwinds123” used as a server password (security researcher warned SolarWinds of this!)

SolarWinds hack (cont.)

Why were basic security practices neglected at SolarWinds? Under-investment in security…

- "Employees say that under [CEO] Mr. Thompson ... every part of the business was examined for cost savings and **common security practices were eschewed because of their expense**. His approach helped almost triple SolarWinds' annual profit margins to more than $453 million in 2019 from $152 million in 2010."
- Bruce Schneier: "The market does not reward security, safety or transparency. It doesn't reward reliability past a bare minimum, and it doesn't reward resilience at all."
- Core problem: Limited economic incentives to invest in cybersecurity
  - Expense with diminishing returns
  - Limited legal liability
  - Small factor in customers' decisions, small effect on share price
  - Negative impact to supply chain security

Investment in security

Gordon-Loeb model

- Even with optimal incentives, firms will never invest more than 37% of expected damage from security breaches in cybersecurity

Reason

- Cybersecurity is not generating profit and having diminishing returns on investment
- If you expect fire damage to cause $10,000 damage, it doesn't make sense to purchase a $10,000 device that reduces the probability of fire damage

Result

- Total damage caused by cybersecurity will always significantly exceed investment in cybersecurity

Legal liability

Having poor cybersecurity is legal

- Limited laws regulating cybersecurity standards

Federal Trade Commission relies on “unfair or deceptive acts” to press charges

Customers and shareholders need extreme cases of negligence or false statements

- Class-action lawsuit against SolarWinds by shareholders only because they allege false and misleading statements

If an honest effort is made, very little legal risk in having bad cybersecurity

Lack of business consequences (or accountability)

Over time, customers tend to forgive and forget data breaches

- Equifax, eBay, Adobe, and Marriott all recovered from their breaches

In corporate context

- Incentives in procurement favor functionality and cost over possible cybersecurity risks
- Difficult to evaluate cybersecurity between companies
- Share prices usually drop heavily after a data breach, but studies show a negligible long-term effect
- More recent data breaches have had smaller share price drops due to “breach fatigue”

Supply chain issues

Centralization of software

- Creates points of vulnerability that dramatically reduce hacker effort
  - SolarWinds allowed hackers to access 18,000 systems
- Vulnerabilities in one company's product have cascading effects
  - Beyond their immediate customers
  - CISA: 30% of SolarWinds victims did not use SolarWinds
- Example: 2017 NotPetya attack
  - Malware deployed by a malicious automatic update in MeDoc (Ukrainian tax preparation software)
  - Caused $10 billion damage
  - Damaged pharmaceutical production, global shipping, hospital systems

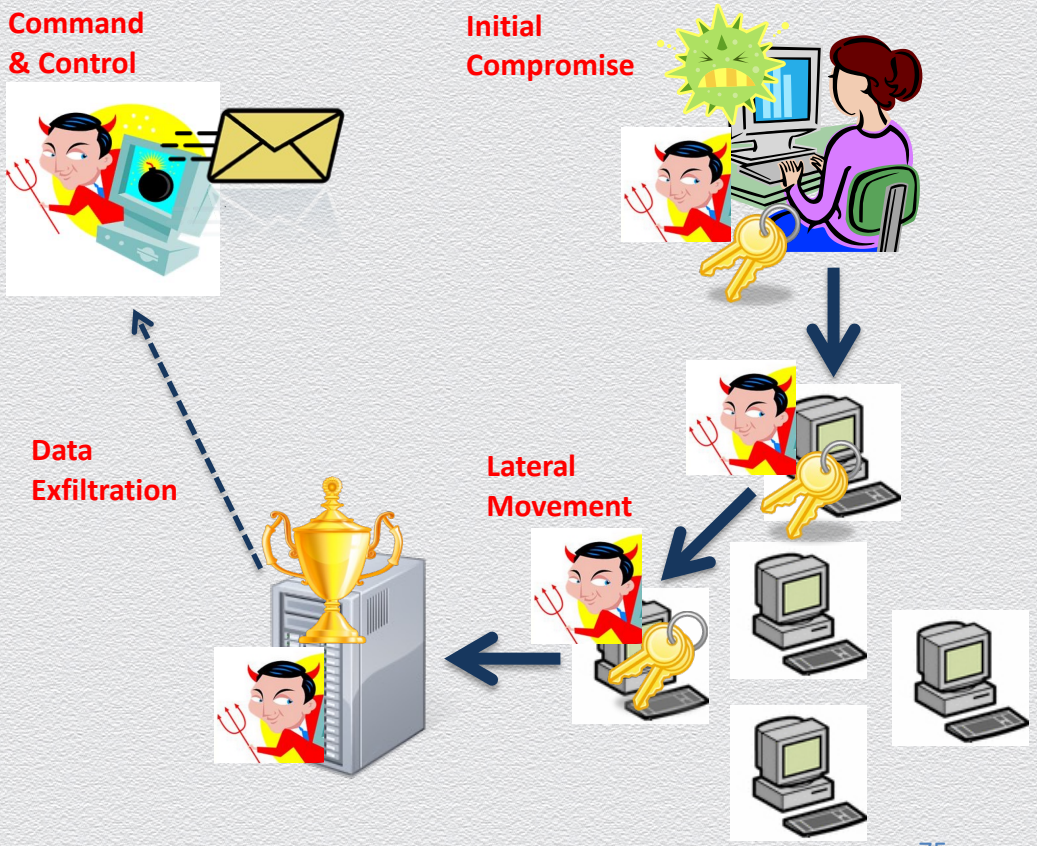
19.6 PillarBox (SIEMs & security analytics)

Advanced Threats: Enterprises' toughest enemy

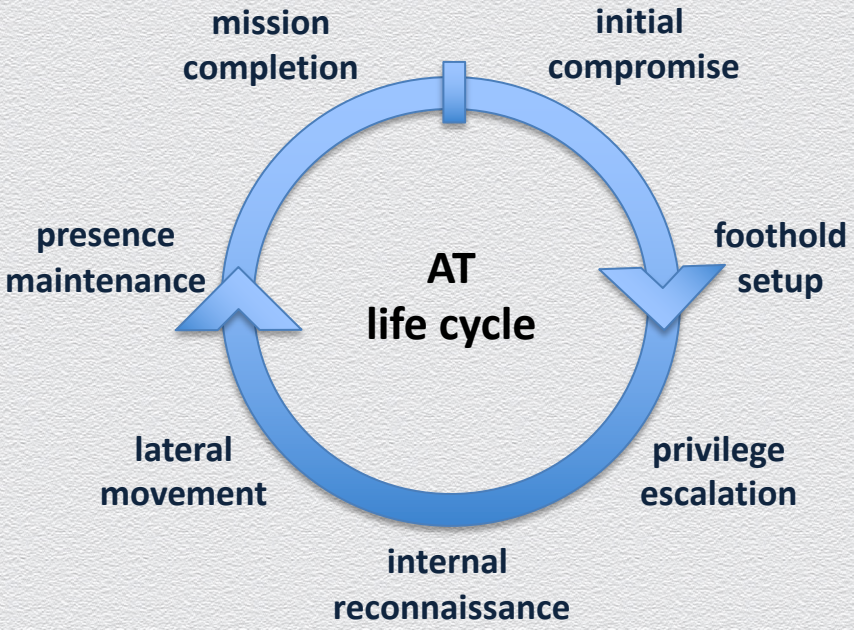
- Advanced Threats (ATs) are a serious risk facing enterprises today
  - comprise well-targeted, persistent attacks
  - aim at unauthorized data manipulation or exfiltration
  - employ rich attack vectors and unknown strategies
    - social engineering
    - zero-day malwares / vulnerabilities
    - low-and-slow progression

➔Extremely hard-to-defend, often even hard-to-detect

The “canonical” attack cycle



The “canonical” attack cycle



Best defenses in security industry

- Tighter preventative practices
  - raise the protection fence
    - e.g., multi-factor authentication, data protections, access control, etc.
- **Detection & forensics tools**
  - visibility – analysis – action
    - e.g., security information event management (SIEM) systems, security analytics

Intelligent-based security

Security Operation Center (SOC)

Command & Control

Initial Compromise

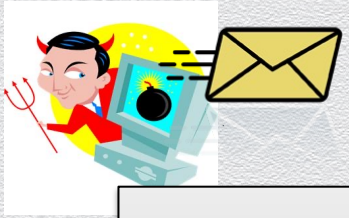
Data Exfiltration

Lateral Movement

Security Information Event Management (SIEM)

Intelligent-based security

Command & Control



Initial Compromise

Security Operation Center (SOC)

Data Exfiltra

Gain visibility into enterprise infrastructure

- monitor complex systems via continuous “health report
- collect device logs and/or special security-related alerts
- correlate received data across different endpoint device
- analyze collected information for incident-respond decision making

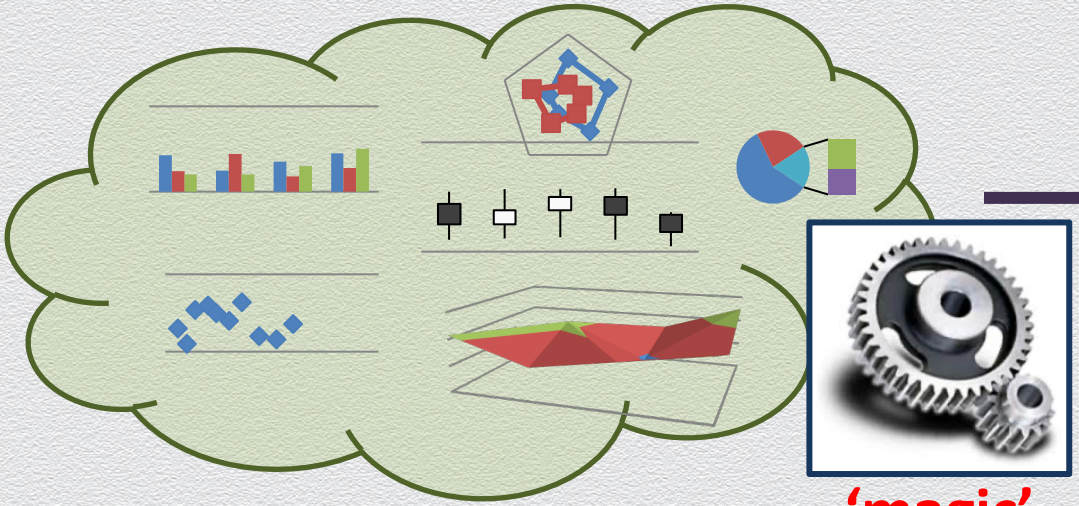
Security Analytics Sources (SASs)

- firewalls, VPNs or endpoint instrumentation, like intrusion-detection systems (IDSs), syslog, anti-virus engine or other alerting facilities

Security Information Event Management (SIEM)

(Big-)Data analytics for security

1. gather evidence from various agents



‘magic’

2. profile normal activities

3. label unusual activities/events

4. detect anomalies

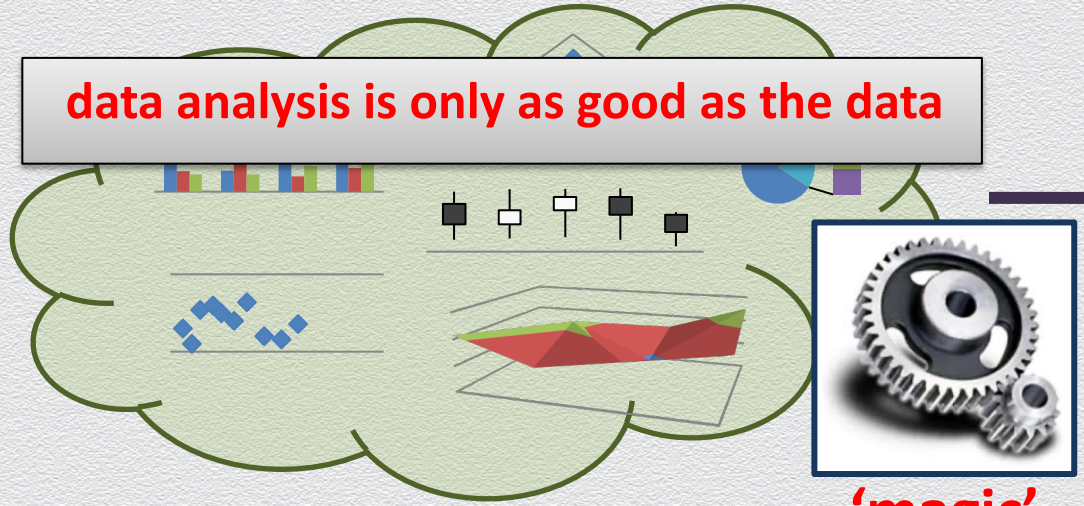
(Big-)Data analytics for security

1. gather evidence from various agents

4. detect anomalies

data analysis is only as good as the data

complex big-data problem



'magic'

2. profile normal activities

3. label unusual activities/events

Next-generation Advanced Threats

- Tougher, evolving adversaries who
  - grow in sophistication to become context aware and target specific
    - know "*what they attack and how it is protected*"
  - shift towards qualitatively stronger attack strategies

Achieve their objective <u>while trying to evade</u> defensive tools

(past / current)

→

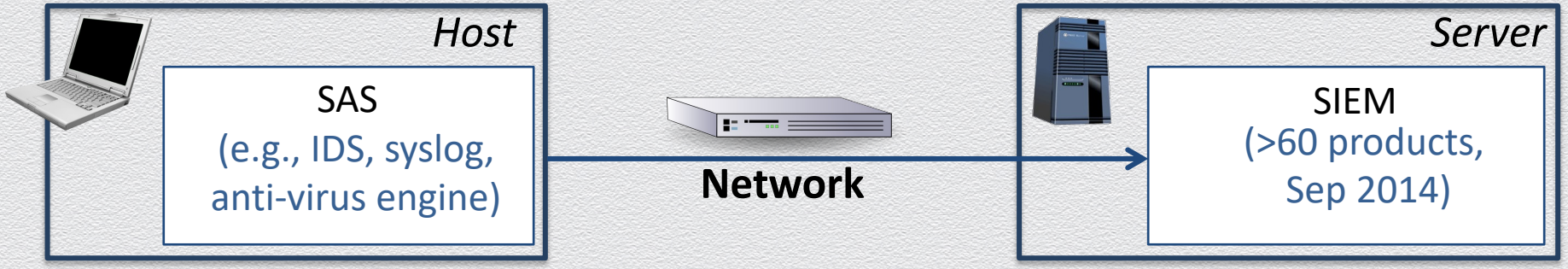
Achieve their objective <u>by first disarming</u> defensive tools

(current / future)

In practice this means…

- If strong authentication is used, the attacker will steal
  - stored keys to clone authenticators
  - passwords to impersonate users
  - credentials to forge signatures
- If security logs are collected and analyzed, the attacker will
  - block the stream of reported logs
  - employ log-scrubbing malware to cover its tracks
  - tamper with host-side log generation software

Problem: Secure chain of custody in security analytics



Problem: Secure chain of custody in security analytics

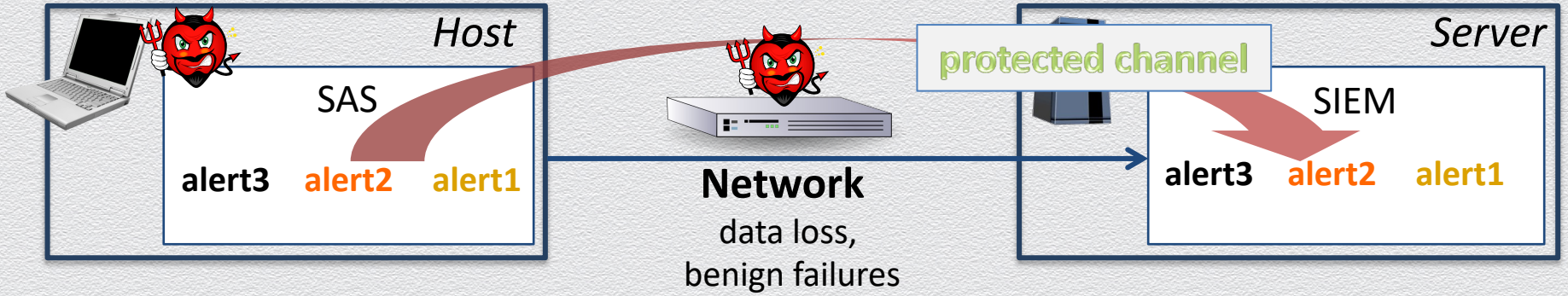
- Security alert systems often employ unreliable channels!

Problem: Secure chain of custody in security analytics

- Security alert systems constitute a direct target of a '2nd-wave' AT!
  - an attacker may discover, observe or read alert transmissions
    - …and accordingly adapt its attack strategy based on SAS behavior!
  - an attacker may tamper, suppress or block alert transmissions
    - …and eventually disrupt SAS functionality (e.g., using log-scrubbing malware)!

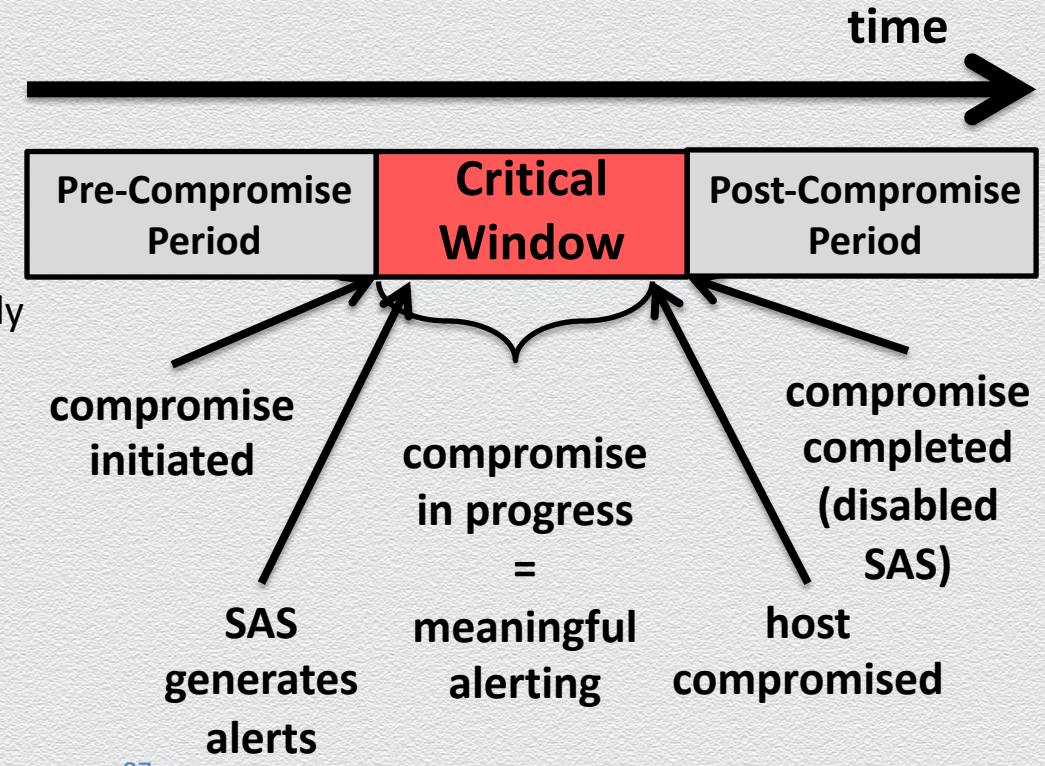
Solution: PillarBox, a secure alert-relaying tool

- Features
  - ensures against alert suppression or tampering
  - conceals alerting activity
  - features self-protection, transmits alerts persistently
  - is agnostic of the exact SAS in use



Which alerts can be relayed and thus protected?

- As a security exploit unfolds on a host, it often produces strong indications visible to resident security software
  - e.g., privilege escalation, registry-entry change, reboot, anti-virus / IDS alert
- A SAS can generate meaningful alerts only while intrusion is in progress
  - e.g., a fully compromised host will automatically shut the SAS down
- We must assume a non-zero critical window of high-indicative alerts

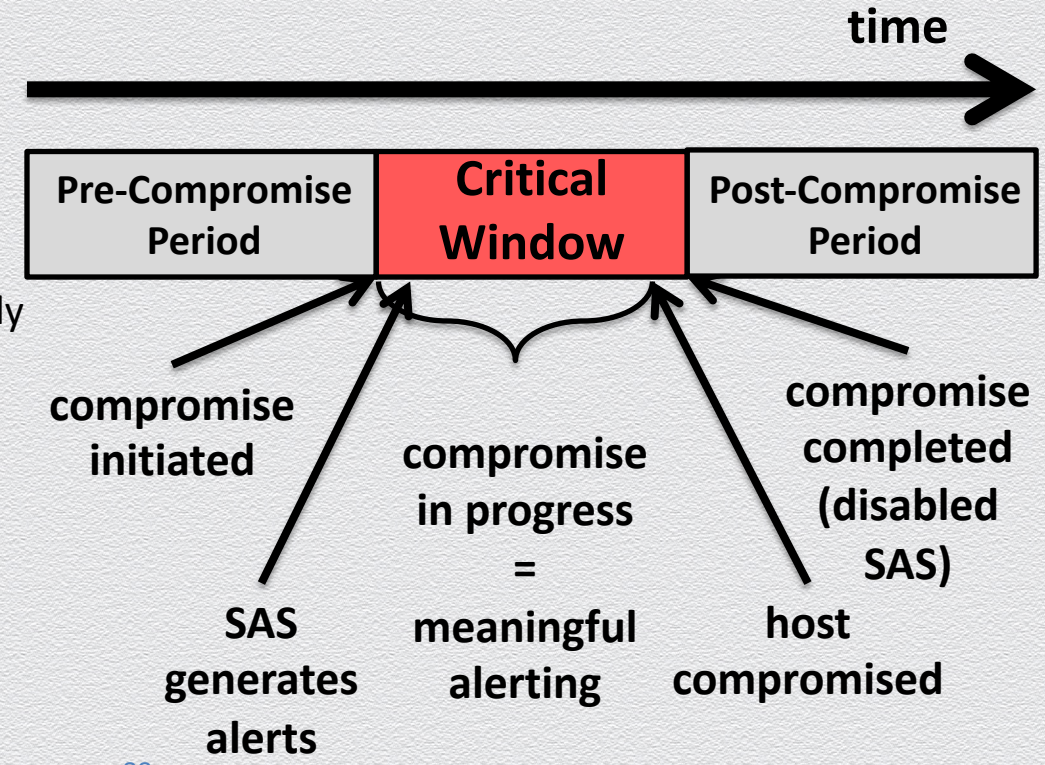


Which alerts can be relayed and thus protected?

- As a security exploit unfolds on a host, it often produces strong indications visible
- nly
- window of high-indicative alerts

race condition between malware & SAS

our goal is to protect alerts **in self-protection** mode, i.e., when SAS is being actively by attacker knowing its operational setting



On-the-fly transmission Vs. buffering of alerts

Push alerts (instantly)

- Unreliable / failed transmissions
  - BYOD hosts, UDP syslog transport
- Network attacks to suppress alerts
  - ARP stack smashing, DoS attack
- SAS intelligence leakage
  - via outbound traffic analysis
- Delayed alert custody
  - prolonged attack window

Buffer alerts

- Persistence
  - alerts are transmitted redundantly
- Regularity / Periodicity
  - in-traffic alert suppression is detected
- Stealth
  - alerting behavior is concealed
- Speed
  - engineered to lock alerts very fast

Vulnerability of on-the-fly alerting

We successfully performed the following attack:

1. use stack overflow exploit in SSH v1 to inject a shell
2. use “Full Nelson” local privilege escalation to gain root access
   - e.g., vulnerabilities CVE-2010-4258, CVE-2010-3849, CVE-2010-3850
3. use automated script to read snort.conf
4. learn if the attack was detectable
5. remove snort log from log file

1. Buffering alerts

(FS) integrity ✓

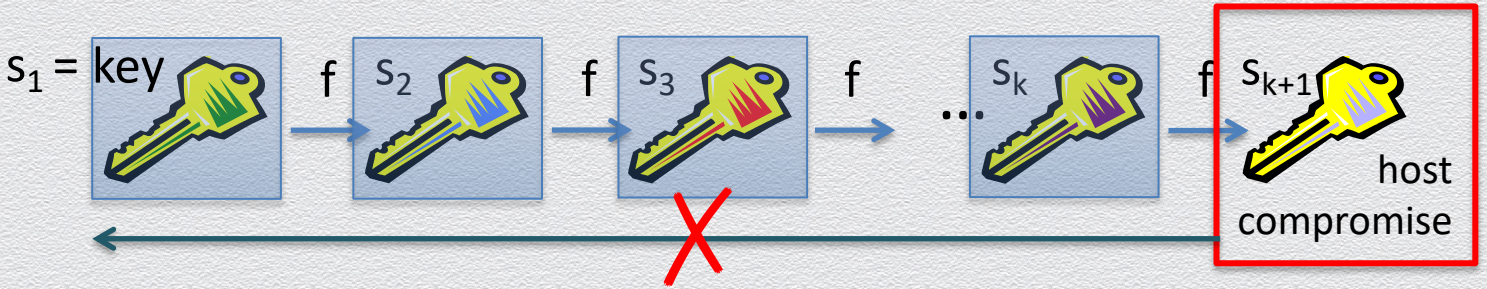
(FS) confidentiality ✓

- As soon as they are generated, alerts are
  - **signed** and **encrypted** using a forward-secure secret key (shared by the server and host) and then stored in a buffer at the host
  - periodically or on demand (e.g., every t alerts) **transferred** to the server

Forward-secure key updates

Attacker can't learn post-compromise cryptographic keys

- update key irreversibly through one-way hashing



2. Retransmitting alerts

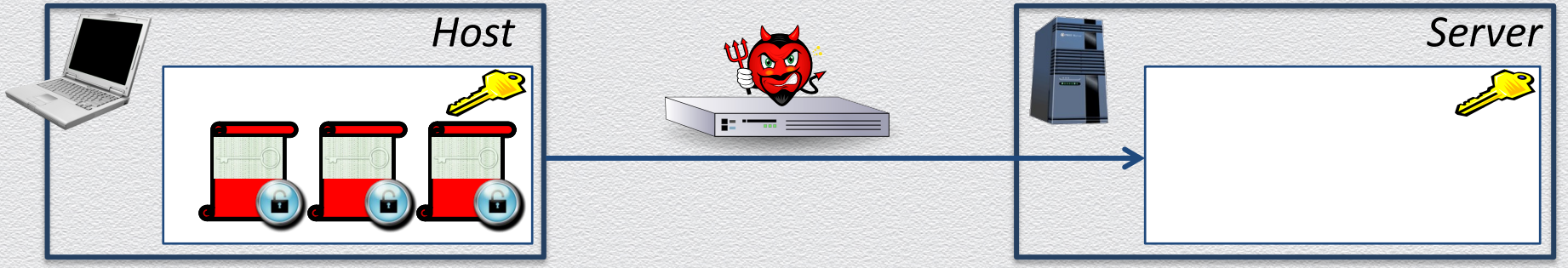
(FS) integrity ✓

(FS) confidentiality ✓

persistence ✓

- As before, but now alerts
  - are **not deleted** from buffer but are **transferred redundantly**
    - e.g., when a new alert is generated all buffered alerts

persistence:
missing alerts can only be attributed to an attack, thus allowing to signal a “meta alert”



2. Retransmitting alerts

(FS) integrity ✓
(FS) confidentiality ✓
persistence ✓

- As before, but now alerts
  - are **not deleted** from buffer but are **transferred redundantly**
    - e.g., when a new alert is generated all buffered alerts are transmitted

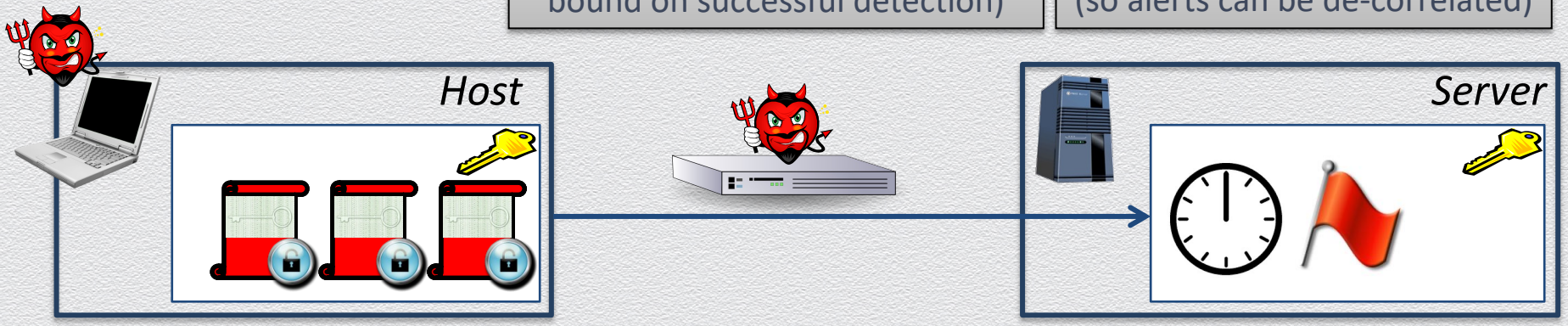
3. Checking heartbeat

(FS) integrity ✓
(FS) confidentiality ✓
persistence ✓
failure detection ✓
traffic concealment ✓

- As before, but now alerts
  - are transmitted **periodically** (in regular time intervals)
  - if failed to reach the

failure detection:
imposes a minimum frequency of transmission (allows an upper bound on successful detection)

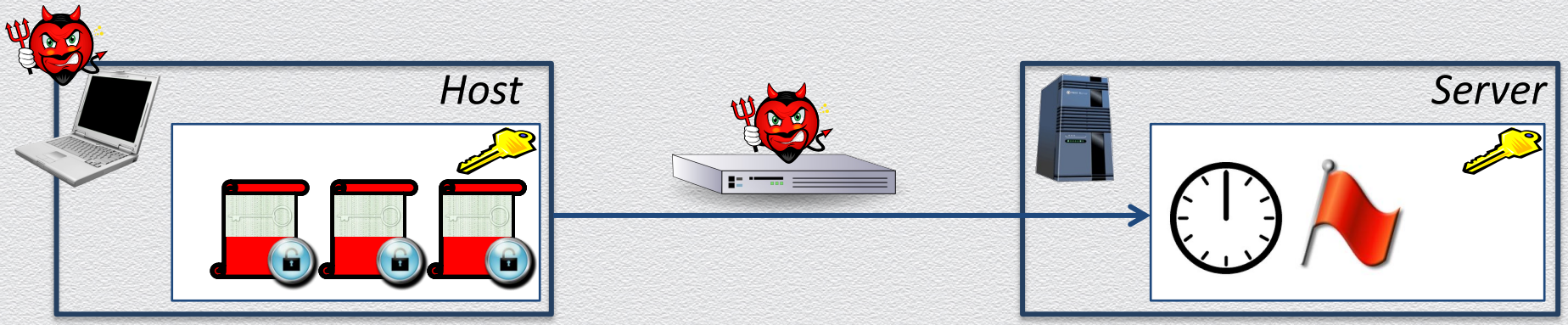
traffic concealment:
imposes a regular pattern of transmissions (so alerts can be de-correlated)



3. Checking heartbeat

(FS) integrity ✓ failure detection ✓
(FS) confidentiality ✓ traffic concealment ✓
persistence ✓

- As before, but now alerts
  - are transmitted **periodically** (in regular time intervals)
  - if failed to reach the server, they signal a **"heartbeat" failure** of SAS

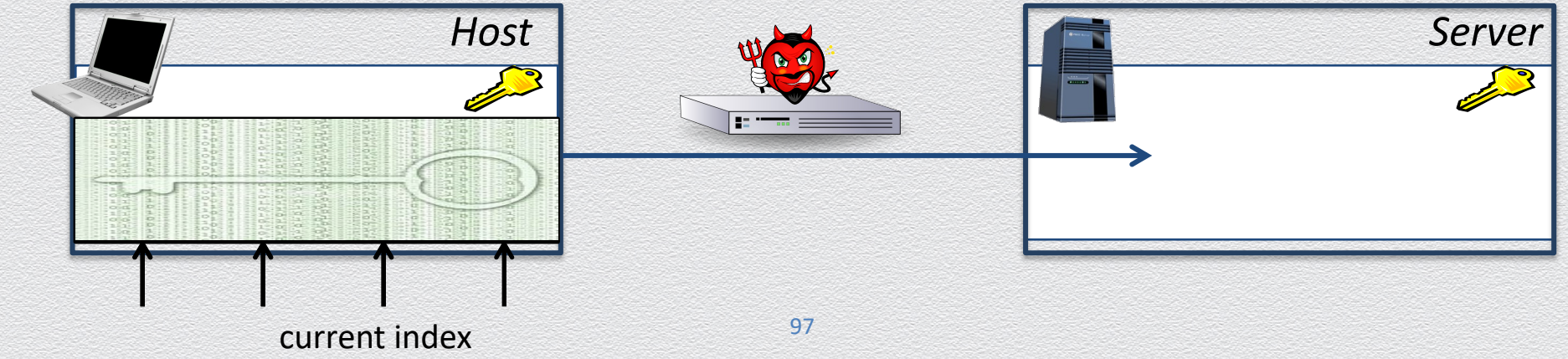


4. Encrypting fixed-size buffers

(FS) integrity ✓ failure detection ✓
(FS) confidentiality ✓ traffic concealment ✓
persistence ✓ stealth ✓

- As before, but now alerts
  - are stored in an initially **random, fixed-size** buffe
  - are transmitted periodically **encrypted** as a whol
  - if failed to reach the server, they signal a **"gap al**

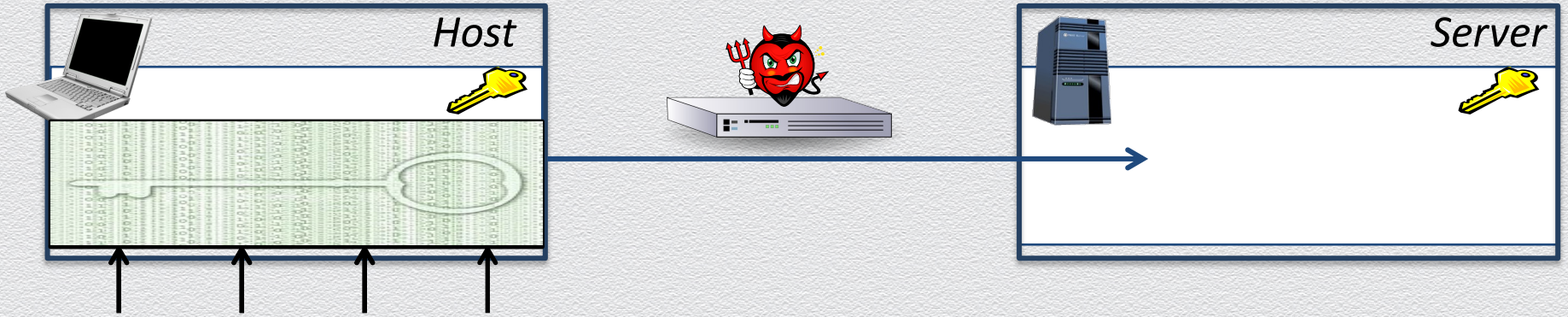
stealth:
alerting mechanism is completely hidden from attacker
(at some communication overhead)



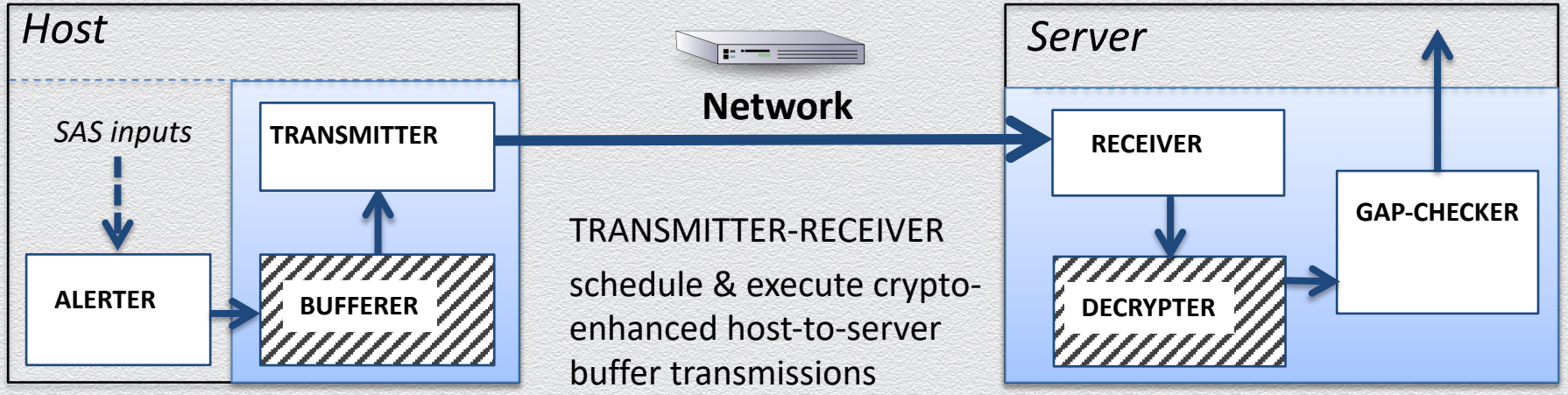
4. Encrypting fixed-size buffers

(FS) integrity ✓ failure detection ✓
(FS) confidentiality ✓ traffic concealment ✓
persistence ✓ stealth ✓

- As before, but now alerts
  - are stored in an initially **random, fixed-size** buffer in a **round-robin** fashion
  - are transmitted periodically **encrypted** as a whole **at the buffer level**
  - if failed to reach the server, they signal a **“gap alert” failure** of SAS



PillarBox architecture



TRANSMITTER-RECEIVER

schedule & execute crypto-enhanced host-to-server buffer transmissions

ALERTER

implements SAS, monitors host to identify events against a set of alert rules, creates alert messages and relays them to BUFFERER

BUFFERER-DECRYPTER

implement crypto-assisted reliable channel & report integrity failures

GAP-CHECKER

reconstructs alert stream, checks for missing alerts, reports “heartbeat” or “gap alert” failures

PillarBox prototype

- Implementation in C++, Inter Xeon E5506, 2.13GHz, quad-core, 4MB L2 cache, 3GB RAM, Red Hat Enterprise Linux WS v5.3 x86_64
- ALERTER: Snort, configured to log events indicating impeding compromise
- BUFFERER: data structure residing in main memory
  - buffer size decided at creation time, dynamically updated on based on network disruptions
  - registers a watch on the log file used by ALERTER to grab new logs
- TRANSMITTER: thread waking after a specified interval decided at run time
  - Authenticated encryption using EAX-mode encryption
  - Custom FS pseudorandom generator for efficient irregular key updates
- GAP-CHECKER: two threads checking heartbeat and gap-alert failures

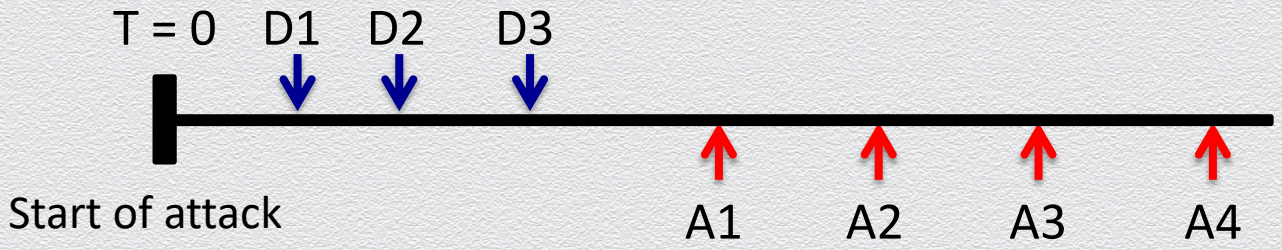
Buffer Vs. on-the-fly alerting

- With PillarBox alerts can be generally locked away in the buffer **faster** than they can be sent over the network to a remote server
  - logs are grabbed from Snort as they are generated and sent directly onto the network using UDP

System	Average	Std. Dev.
PillarBox alert locking	163.06µs	49.09µs
onto-the-wire alert transmission	251.78µs	41.00µs

Race condition: ideal ordering

Non-zero critical window is necessary for PillarBox to be effective



D1 : Attack detected and logged

D2 : Attack secured by PillarBox

D3 : Triggered alert rule deleted

A1 : Remote attack completes

A2 : Privilege escalation

A3 : Attacker learns detection rules

A4 : Attacker deleted logged file

Race condition: results

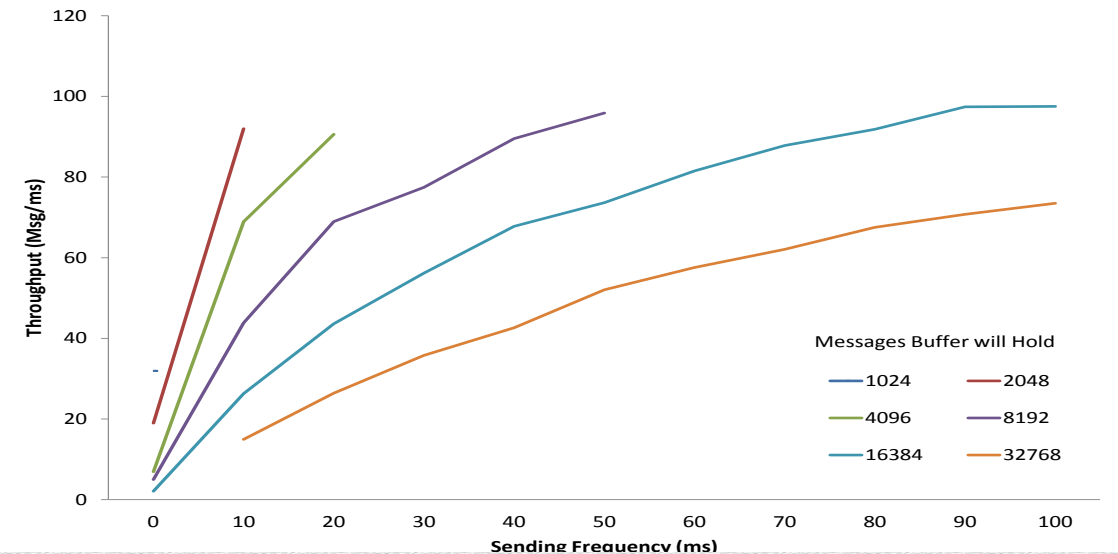
PillarBox consuming alerts from Snort can secure alerts before full host compromise

Event	Average (µs)	Std. Dev. (µs)
D1 : Attack detected and logged	1,645.441	1,069.843
D2 : Attack secured by PillarBox	1,645.609	1,069.842
D3 : Triggered alert rule deleted	1,645.772	1,069.840
A1 : Remote attack completes	2,692.536	1,324.419
A2 : Privilege escalation	2,693.474	1,324.432
A3 : Attacker learns detection rules	2,696.524	1,324.919
A4 : Attacker deleted logged file	2,696.590	1,324.990

Overhead and throughput

PillarBox adds an additional 7% overhead compared to running Snort

PillarBox is fast enough that prevent it from being a bottleneck



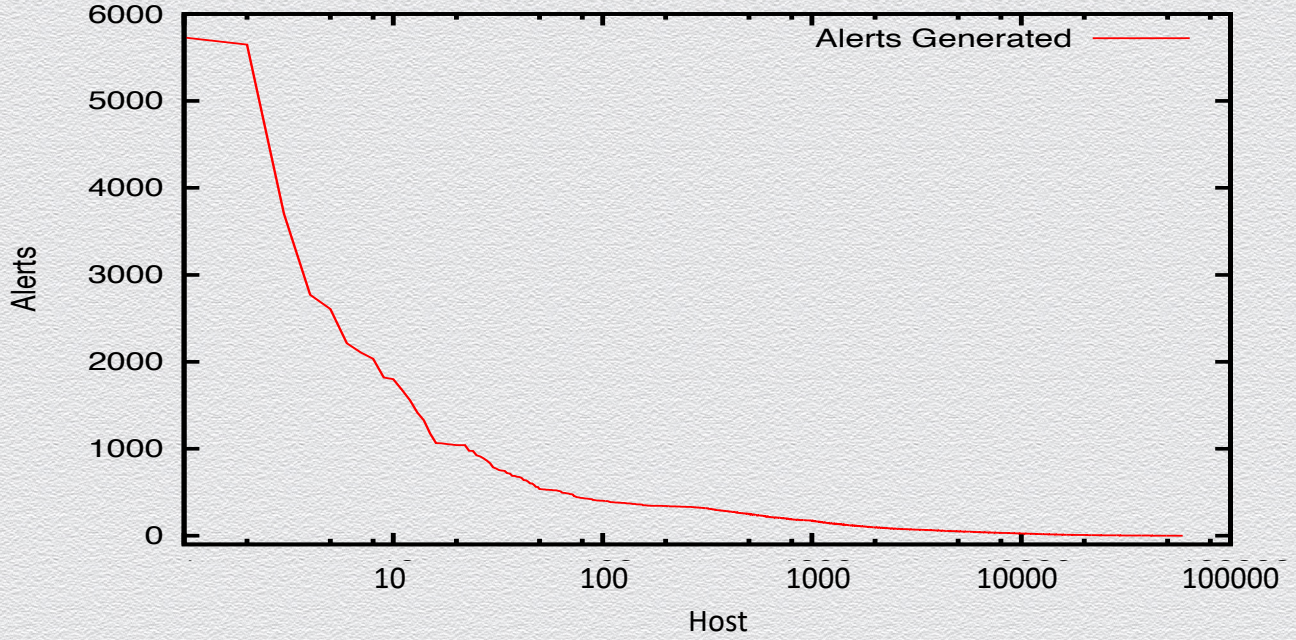
able to process nearly 100,000 alert messages per second,
well above the recording rate achievable by Snort

typical Snort alert messages (a few hundred characters)

Observed alerting frequencies

Analysis of a large enterprise dataset (59,034 users) across period of 7h:

busiest machine: 8603 alerts; average across all machines 18.3 alerts



by designing PillarBox to handle a throughput of 1 alert per second (i.e., 3600 alerts per hour), we can handle the busiest ALERTER (e.g., 1707 alerts / hour in our dataset)

Summary

- PillarBox is a general-purpose reliable alert relaying tool, offering intrusion-resilient security in log protection
  - buffering, fast forward-secure logging, secure sequencing
- Features
  - self-protection mode of operation
  - integrity, stealth, persistence