CS459/698 Privacy, Cryptography, Network and Data Security

Network Steganography and Information Hiding

Spring 2025, Monday/Wednesday 2:30pm-3:50pm

Definitions

Steganography

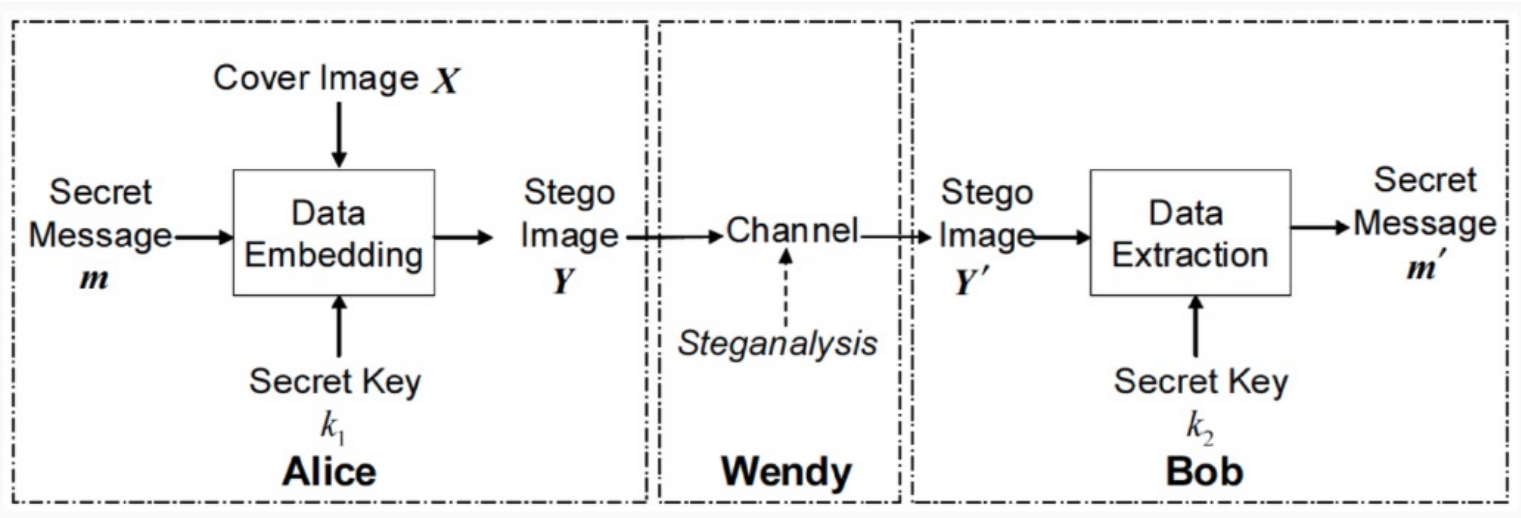
- Art and science of communicating in a way that hides the existence of a message
  - From the Greek words *steganos* and *graphy*

- Steganography takes one piece of (*secret*) information and hides it within another (*carrier / cover*)

Cryptography vs. Steganography

- **Cryptography:** protects the contents of messages
- **Steganography:** conceals the existence of messages

Steganography system model



- Wendy can be seen as a warden, and can be:

  Passive: attempts to detect whether Y carries secret content

  Active: modifies stego image Y into Y' in hopes of destroying the secret content

Why are we studying covert channels?

- Transfer sensitive/unauthorized information through a channel that is not supposed to transmit that information
  - Makes it more difficult to detect data exchanges

Croissant-based covert channel

Why should we care?

- Corporate espionage
- Government or military activities
- Criminal activities
- Censorship circumvention

Covert channel

- A covert channel is a path for the illegal flow of information between subjects within a system, utilizing system resources that were not designed to be used for inter-subject communication.

- What information can be transmitted through a channel may be determined by a policy, physical limitations, etc.

Types of covert channel

- Several dimensions to be considered:
  - Local vs. remote
  - Storage vs. timing
  - Noisy vs. noiseless

- Important characteristics:
  - Bandwidth: how many Bps can be transmitted through the covert channel?
  - Noise: Is the information transmitted through the covert channel distorted in any way?

Local vs. remote covert channels

- Local covert channels leverage a machine's shared resources:
  - CPU, RAM, Disk...

- Remote covert channels leverage transmission mechanisms
  - Typically the network (but also others...)

ETHERLED: Air-gapped systems leak data via network card LEDs

By Bill Toulas August 23, 2022 07:28 AM

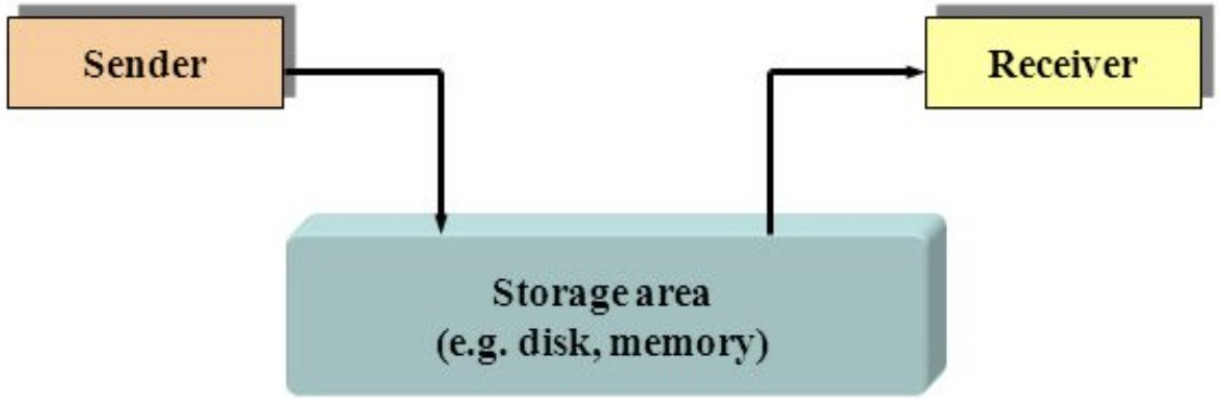
Storage vs. timing covert channels

- **Storage channel:** the sending process alters a particular data item, and the receiving process detects and interprets the value of the altered data to receive information covertly.

- **Timing channel:** the sending process modulates the amount of time required for the receiving process to perform a task or detect a change in an attribute, and the receiving process interprets this delay or lack of delay as information.

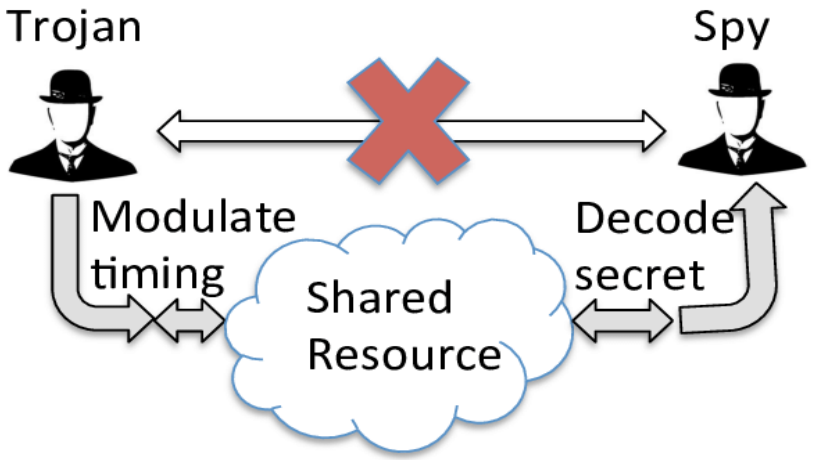
Covert storage channels

- To use a covert storage channel:
  - Both sender and receiver must have access to some attribute of a shared object.
  - The sender must be able to modify the attribute.
  - The receiver must be able to view that attribute
  - A mechanism must be in place for initiating the sender and receiver processes, and there must be a way to sequence their accesses to the shared resource (e.g., sync header)



Covert timing channels

- To use a covert timing channel:
  - Both sender and receiver must have access to some attribute of a shared object.
  - Both sender and receiver have access to a time reference (real-time clock, timer, events order).
  - The sender must be able to control the timing of the detection of a change in the attribute of the receiver.



Can’t we just get rid of covert channels?

- It is typically infeasible to eliminate every potential covert channel in a (networked) computer system, but we can:
  - Eliminate them by modifying the system implementation.
  - Reduce their bandwidth by introducing noise into the channel.
  - Monitor for usage patterns that indicate someone is trying to exploit a covert channel.

Some attempts at detection

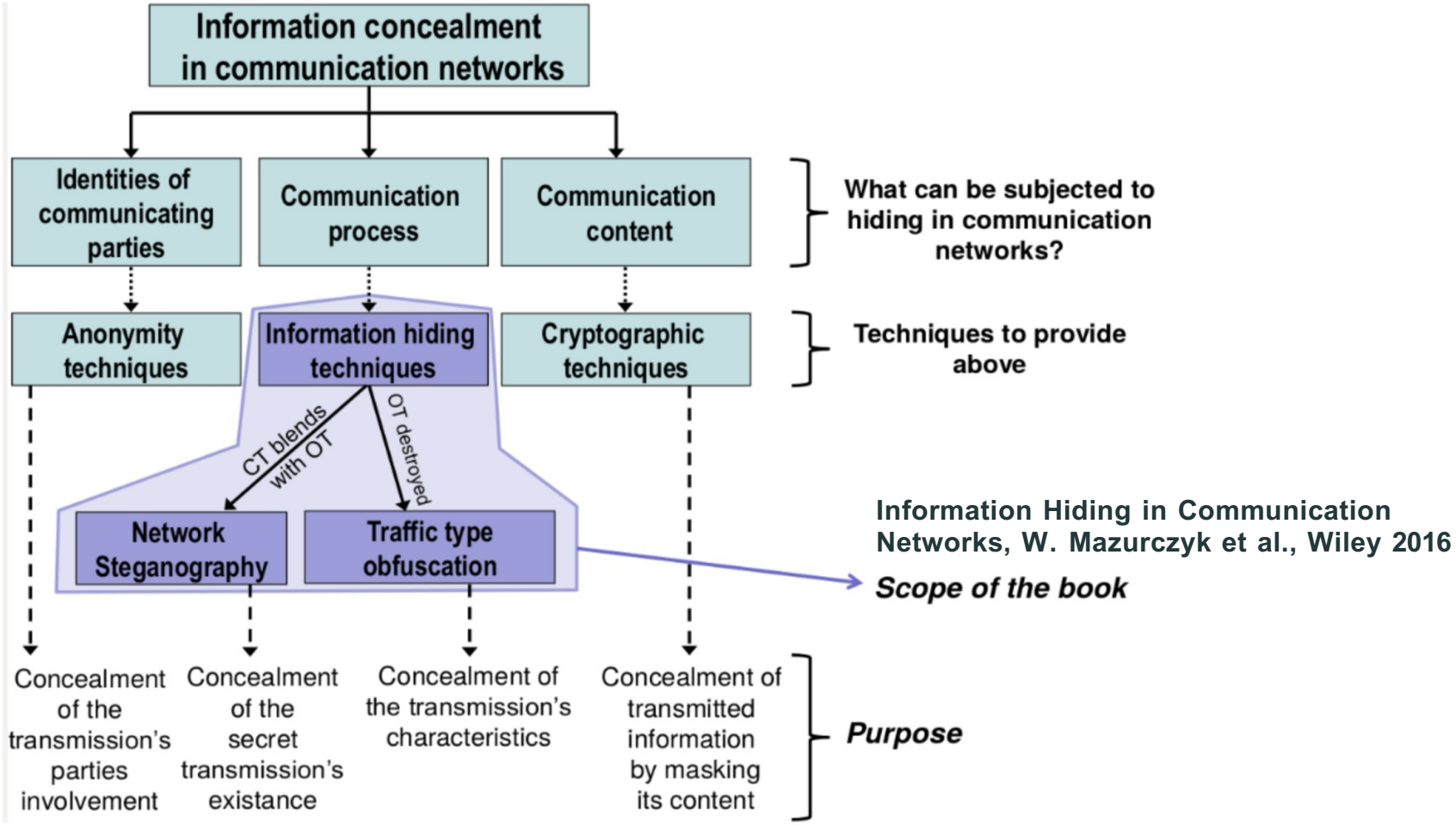
- Kemmerer's Shared Resource Matrix
  - Systematic way to investigate potential covert channels
    - Enumerate shared resources that can be referenced or modified by a subject (i.e., process)
    - Determine whether a given primitive may modify or reference the attribute
  - Requires substantial knowledge about the semantics and implementation of system operations.

RESOURCE ATTRIBUTE \ PRIMITIVE		WRITE FILE	READ FILE	LOCK FILE	UNLOCK FILE	OPEN FILE	CLOSE FILE	FILE LOCKED	FILE OPENED	PROCESS SLEEP
PROCESS	ID									
	ACCESS RIGHTS	R	R	R	R	R	R	R	R	
	BUFFER	R	R,M							
FILES	ID									
	SECURITY CLASSES	R	R	R	R	R	R	R	R	
	LOCKED BY	R	R	R,M	R	R	R	R	R	
	LOCKED	R	R	R,M	R,M	R	R	R	R	
	IN-USE SET	R	R	R	R	R,M	R,M	R	R	
	VALUE	R,M	R							
CURRENT PROCESS		R	R	R	R	R	R	R	R	R,M
SYSTEM CLOCK		R	R	R	R	R	R	R	R	R

Shared resource matrix methodology: an approach to identifying storage and timing channels [Richard Kemmerer, ACM TOCS'83]

Network Information Hiding

Information hiding in the network



Network Information Hiding

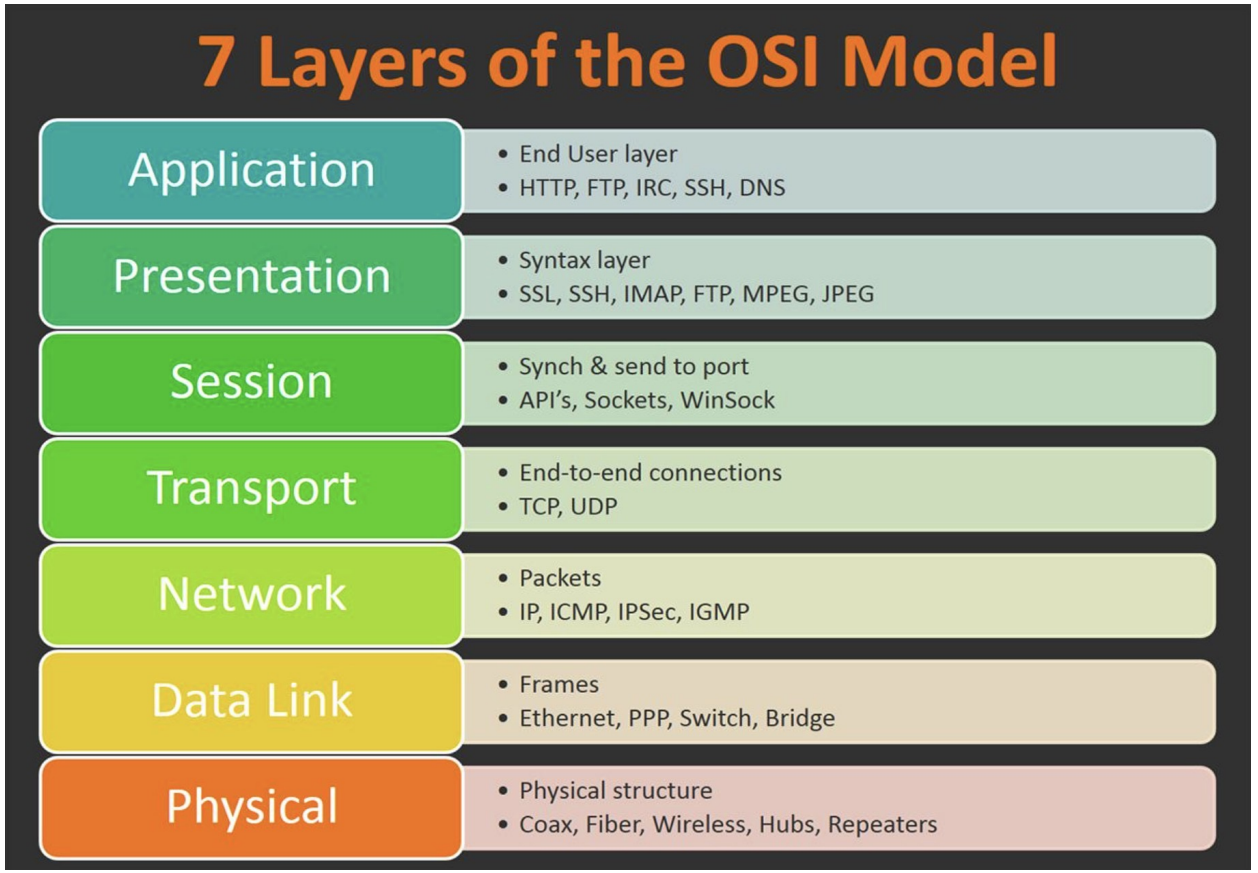
Network covert channels

How do we create a network covert channel?

- Storage
  - e.g., packet header manipulations
- Timing
  - e.g., timing between packets
- What about steganography?
  - We may say that steganographic methods are used to create a network covert channel
- In a network covert channel:
  - Covert data is hidden in overt network transmissions
  - The “cover” medium is called a “carrier”

OSI Layers

- We can implement covert channels across the OSI stack

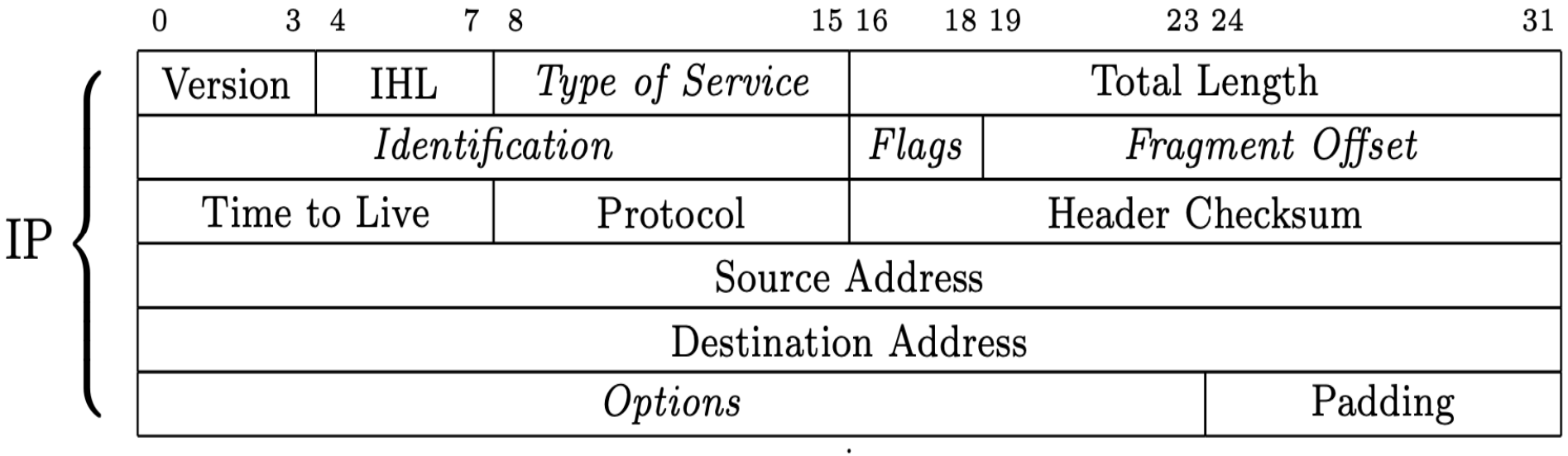


Covert storage channels on TCP/IP

- TCP/IP packets have headers that provide extra information
  - Headers have different fields that are optional or disregarded in usual transmissions

- These fields can be used for hiding information!
  - IP identification
  - Offset
  - Options
  - TCP Checksum
  - TCP Sequence Numbers

IP Header


0 3 4 7 8 15 16 18 19 23 24 31
IP
Version
IHL
Type of Service
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
Options
Padding

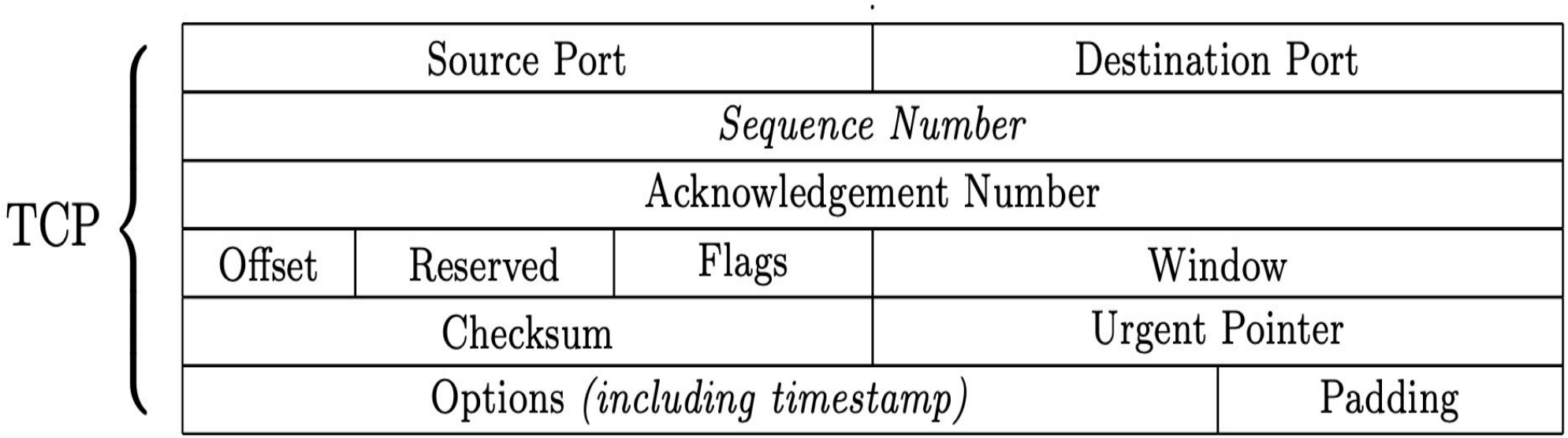
Covert storage channels on IP

- IP ID: a value assigned by the sender to aid in assembling a packet's fragments

- Detection approaches:
  - OpenBSD toggles the most significant bit of the IP ID every 3 minutes or 30;000 IP IDs, so the MSB can be examined to check if it matches this pattern.
  - Within a rekey interval, the OpenBSD IP ID is nonrepeating

Embedding Covert Channels into TCP/IP, Murdoch and Lewis, International Workshop on Information Hiding, 2005

TCP Header


TCP
Source Port
Destination Port
Sequence Number
Acknowledgement Number
Offset
Reserved
Flags
Window
Checksum
Urgent Pointer
Options (including timestamp)
Padding

Covert storage channels on TCP/IP

- TCP ISN: initial sequence number on TCP connections

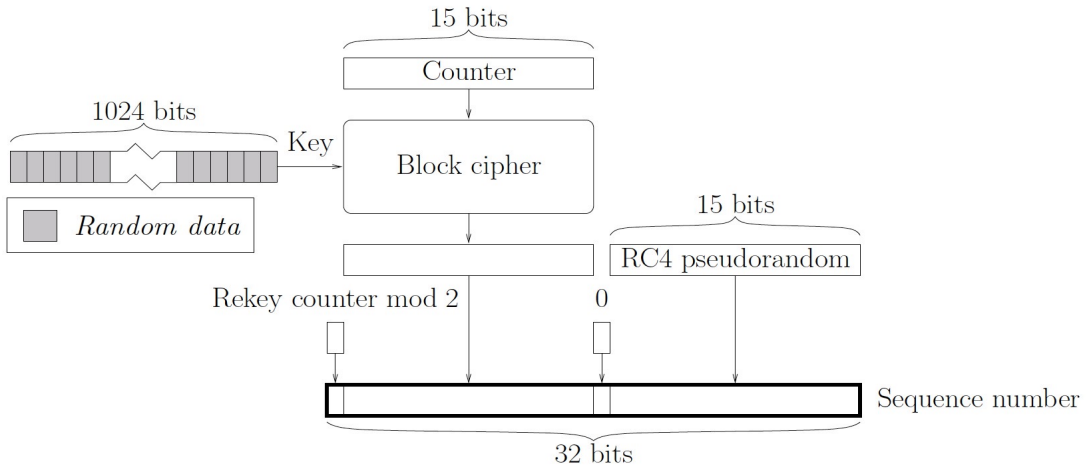
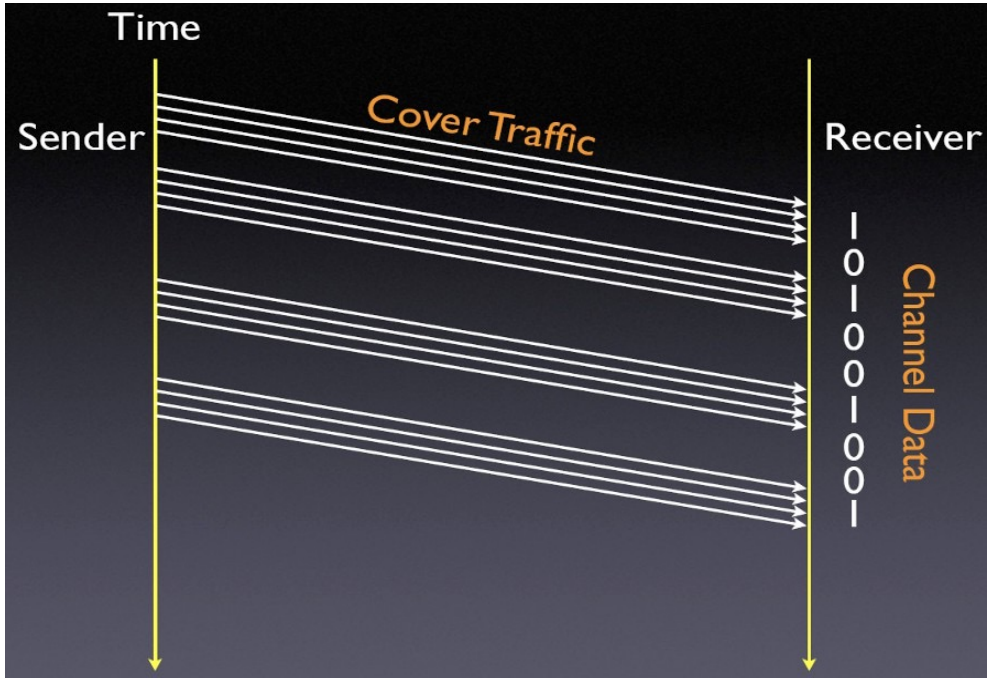


Fig. 4. OpenBSD ISN generator

- Several constraints make steganography easily detectable
- Embedding Covert Channels into TCP/IP, Murdoch and Lewis, IWIH, 2005

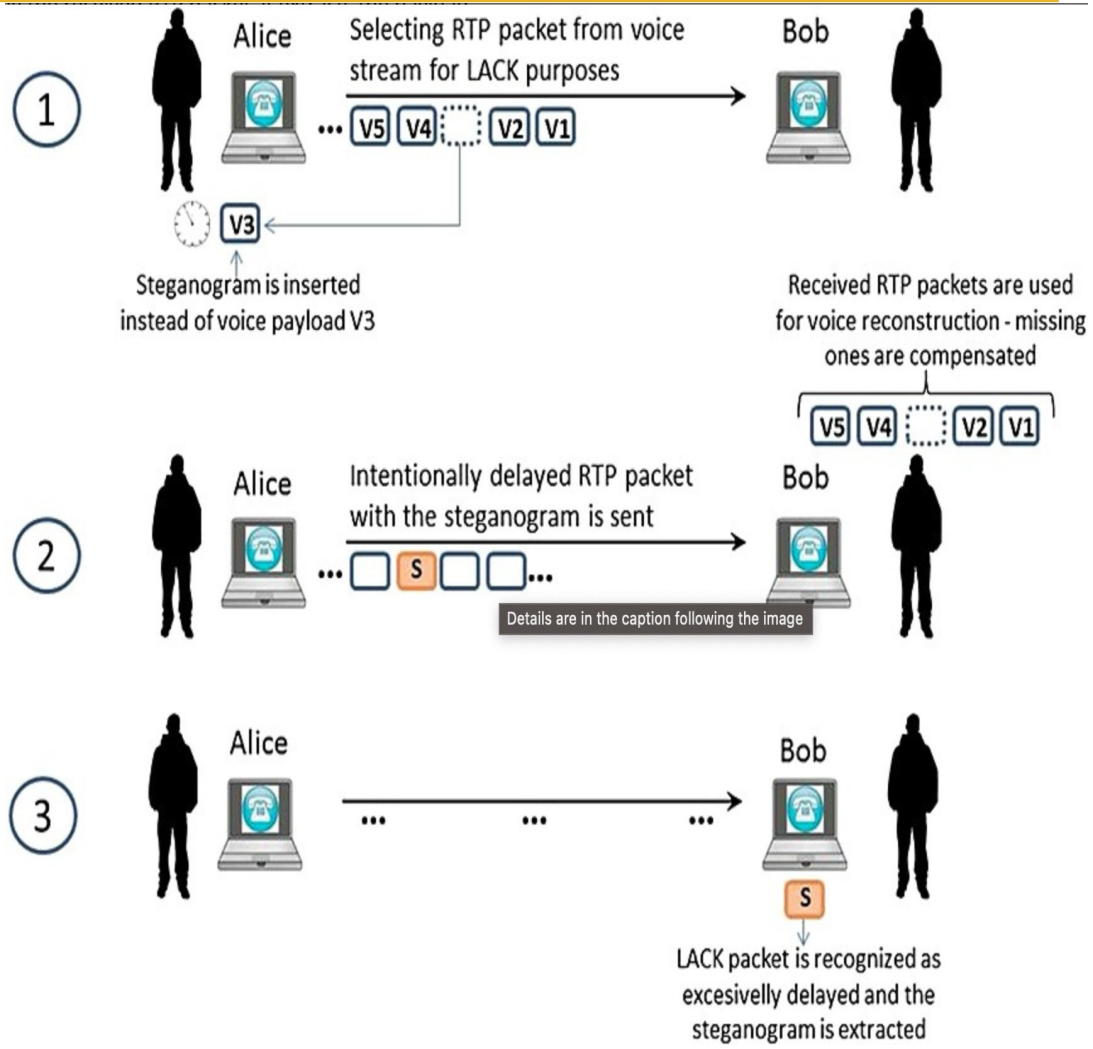
Covert timing channels on TCP/IP

- These typically propagate covert information by crafting delays between certain events
  - e.g., modify usual inter-packet delay, introduce losses by skipping sequence numbers



Covert storage & timing channels on TCP/IP

- We may also have hybrids of storage and timing (e.g., LACK)
  - Replace encrypted packet contents with covert data and use delays for signalling the receiver about specific packets

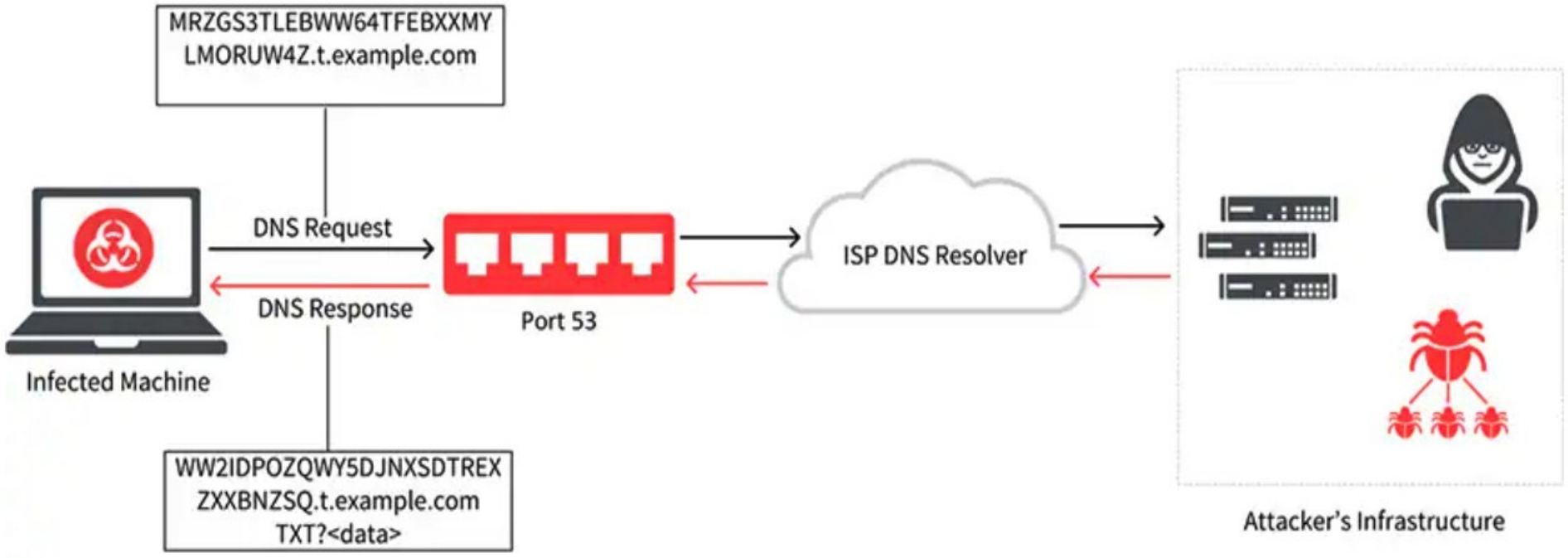


Covert channels at the application level

- Many examples:
  - HTTP
  - DNS
  - Games
  - VoIP/video traffic
  - Push notifications
  - ...

Example: DNS Tunneling

- DNS Tunneling is based on encoding the data of other programs or protocols in DNS queries and responses



Example: Games

- We can create covert channels by encoding information in games' virtual worlds which are shared by multiple users

How to detect/prevent network covert channels

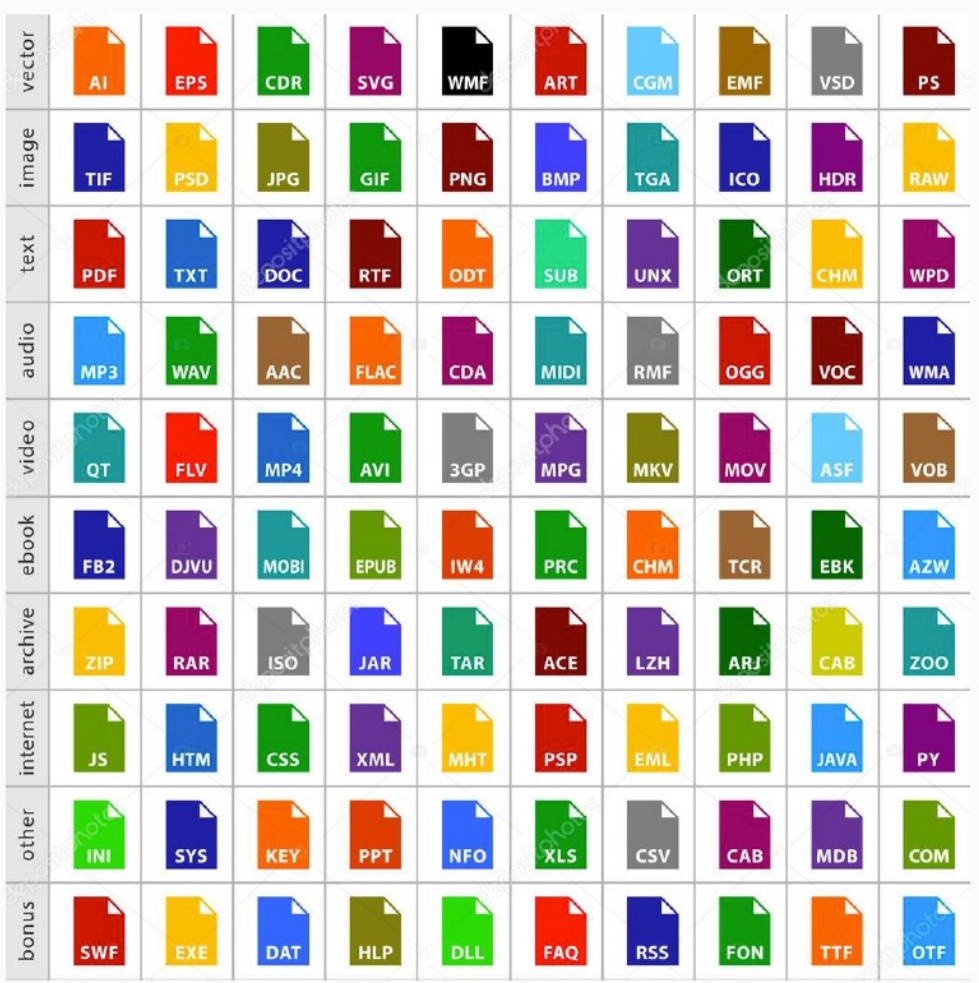
- A warden inspects (and/or manipulates) traffic to detect (and/or break) covert channels
- Storage channels
  - Passive: Analyze transmitted data for anomalies.
  - Active: Normalize data in header fields
- Timing channels
  - Passive: Analyze packet timing for inconsistencies
  - Active: Shape traffic (e.g., constant rate)

File Formats (and a little help for A2)

A Primer on File Formats

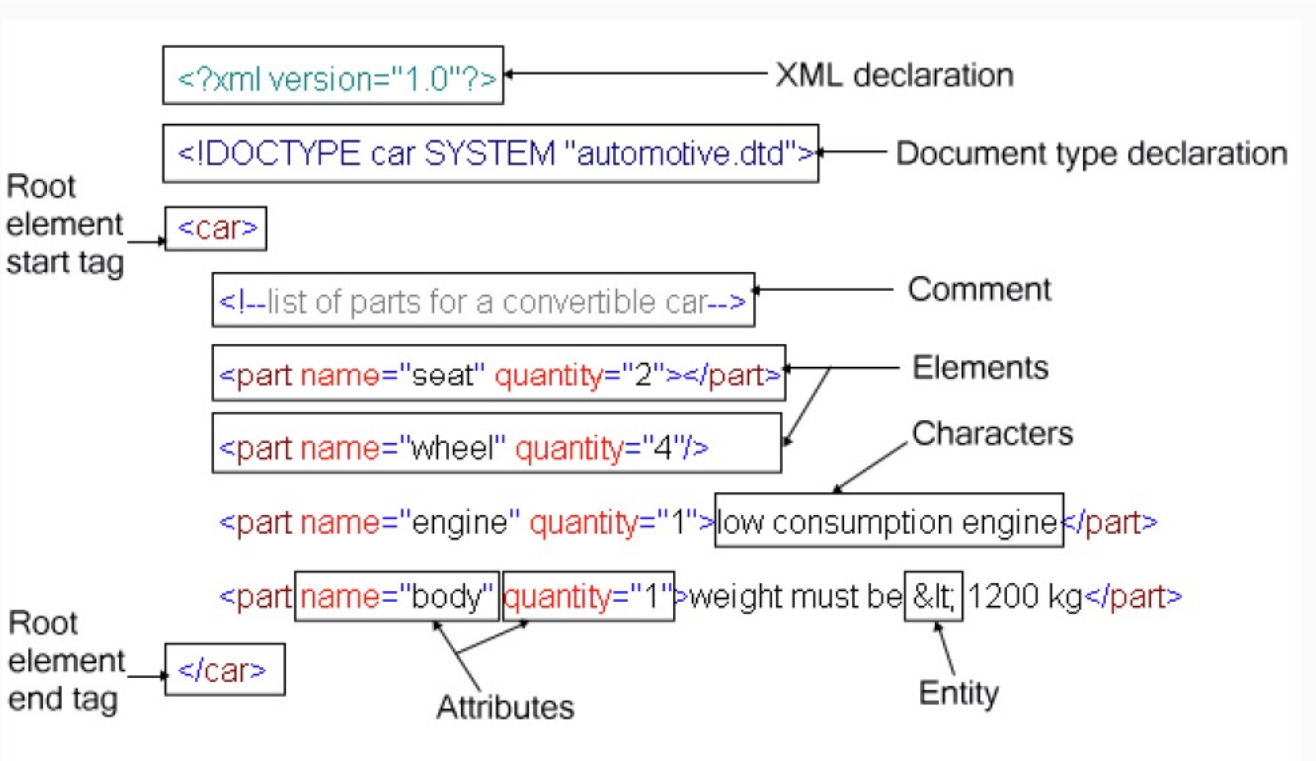
- A file format is a standard way that information is encoded for storage in a computer file

- There are two broad file format families:
  - Text files: Essential to determine the text encoding scheme and structure (if any)
  - Binary files: Essential to determine the file format



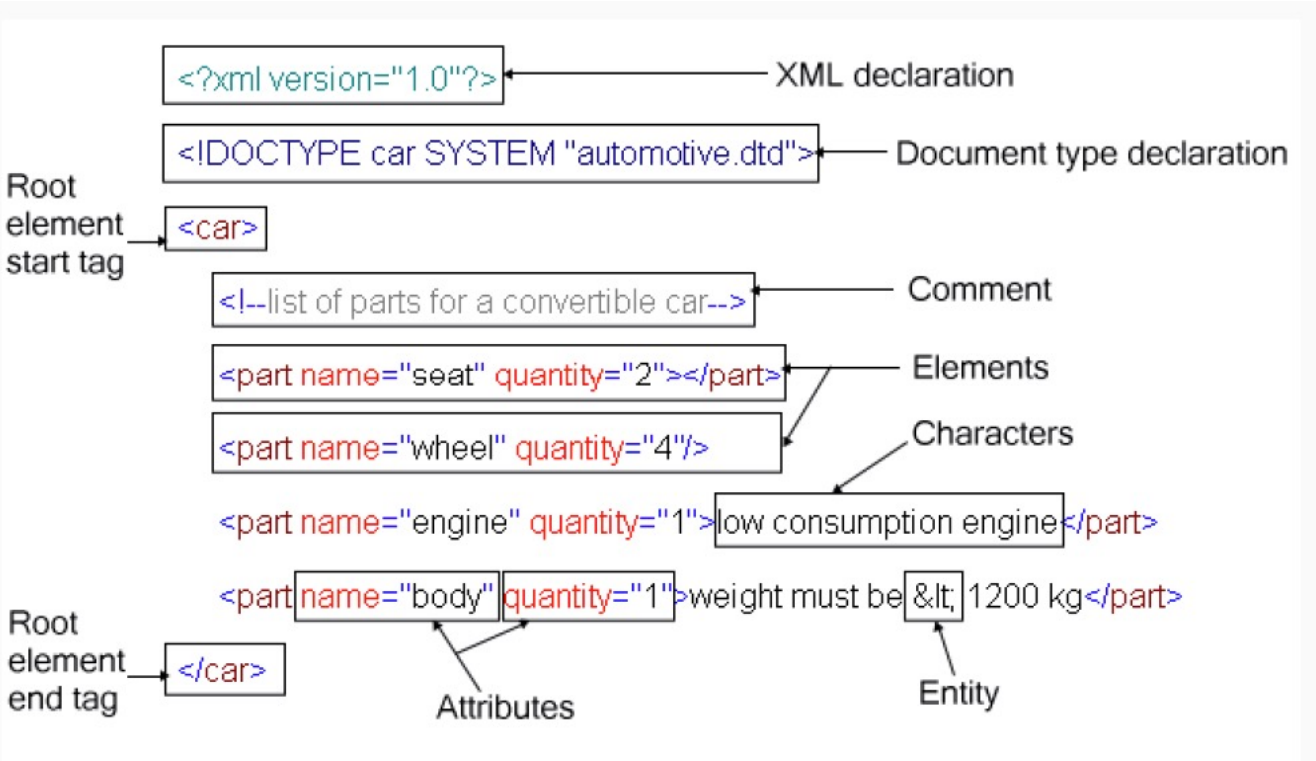
Text Files

- Text files can have some structure on their own
  - E.g., XML, HTML, JSON, etc.



Text Files

- Text files can have some structure on their own
  - E.g., XML, HTML, JSON, etc.



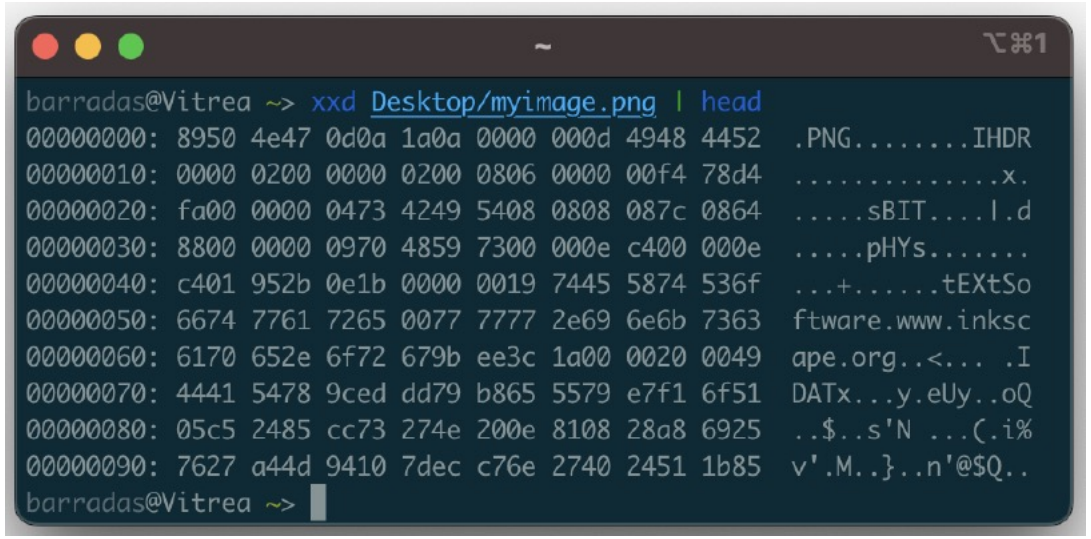
Some of these elements may be used to store covert data as part of a covert storage channel…

Binary Files

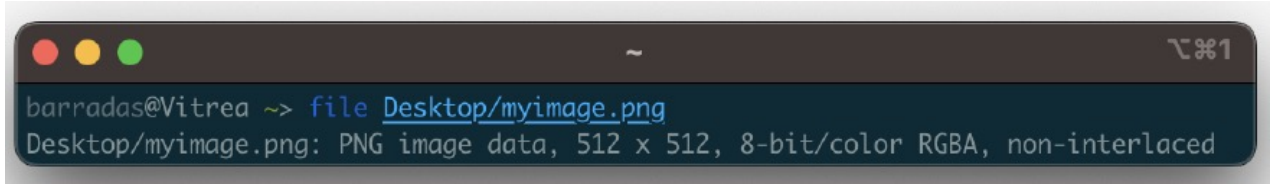
- In binary files, bytes represent custom data
- Binary file formats may include multiple types of data in the same file, such as image, video, and audio data
  - This data can be interpreted by supporting programs, but will show up as garbled text in a text editor

Inspection of a file’s raw bytes

- Use an hex editor to read file contents, e.g., xxd

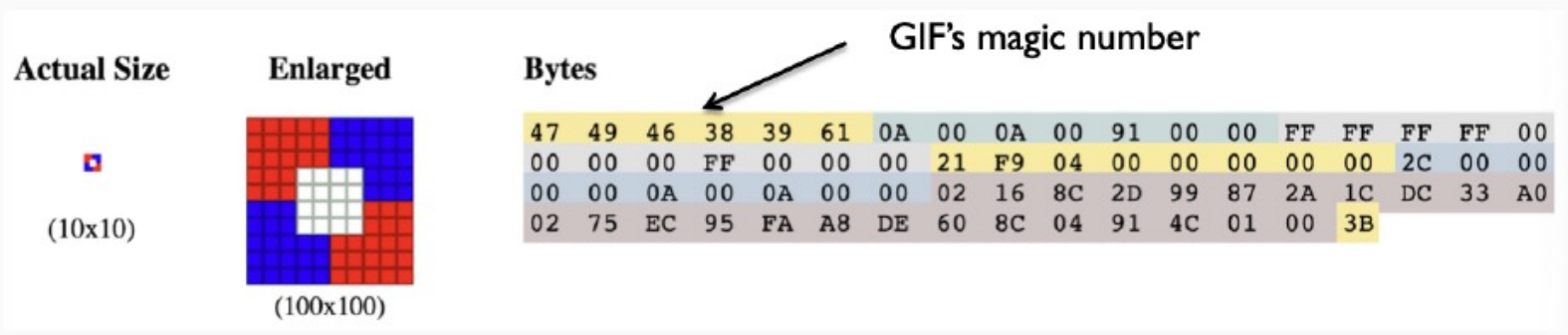


- Use the file utility to match a file’s signature



Magic Numbers

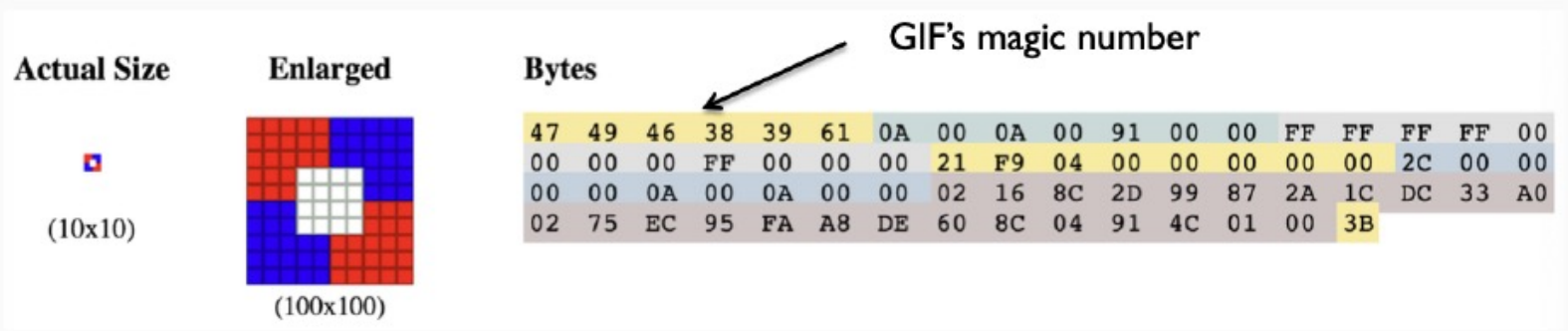
- When in doubt, look for magic numbers
  - Numerical/text values used to identify a file or protocol
  - E.g., GIF files start with the sequence 0x47 49 46 38 39 61



- Magic numbers of common file formats:
- http://www.garykessler.net/library/file_sigs.html

Magic Numbers

- When in doubt, look for magic numbers
  - Numerical/text values used to identify a file or protocol
  - E.g., GIF files start with the sequence 0x47 49 46 38 39 61



- Magic numbers of common file formats:
- http://www.garykessler.net/library/file_sigs.html

Maybe I can use this to make sense out of what's being transmitted within a covert storage channel…

Network Information Hiding

Traffic obfuscation

Information concealment in networks

- Timing and content anomalies may be an effective way to detect covert channels

- Are there better ways to hide the existence of covert data transmissions?

Information concealment in networks

- Well, yes!

- Traffic obfuscation:
  - Hide the characteristics of a covert data transmission by shaping the “look” of data exchanges
  - e.g., used to hide malware communication with a C&C server, evade censorship, etc.

Different techniques for traffic obfuscation

- Randomize traffic
  - Don’t look like any particular protocol
- Mimic traffic
  - Attempt to look like some other protocol
- Tunnel traffic
  - Piggyback on another protocol’s execution

Traffic randomization

- Idea: evade inspection by generating traffic that does not conform to any known protocol specification
  - Randomize packet sizes and timings
  - Randomize packet contents (no signatures)

- Examples:
  - Shadowsocks
  - V2Ray
  - OutlineVPN

Issues with traffic randomization systems

- “Look-like-nothing” might be a signature in itself
- Does not work if wardens have protocol allowlists in place
- Can be detected via cryptographic flaws and entropy tests
  - Security Notions for Fully Encrypted Protocols [Fenske and Johnson. FOCI’23]
  - How the Great Firewall of China Detects and Blocks Fully Encrypted Traffic [Wu et al., USENIX Security’23]

Traffic mimicking

- Idea: Hide a protocol's execution by mimicking another innocuous protocol's characteristics (e.g., Skype)
  - Leverage steganography or encrypted carrier protocols
  - Embed covert data in specific protocol fields
  - Mimic how an encrypted cover protocol sends its traffic
- Examples:
  - SkypeMorph [Mohajeri Moghaddam et al. CCS'12]
  - StegoTorus [Weinberg et al. CCS'12]
  - CensorSpoofer [Wang et al. CCS'12]

Issues with traffic mimicking systems

- It is very difficult to build a perfect imitation
  - Respond to network perturbations
  - Cover all corner cases and error conditions (and bugs!)
  - Mimic relationships between sub-protocols
  - Keep up with the cover protocol's updates

- Now believed to be a fundamentally flawed approach
  - The Parrot is Dead: Observing Unobservable Network Communications [Houmansadr et al., S&P'13]

Traffic tunneling

- Idea: Piggyback covert data on the execution of a protocol
  - Send covert data as the protocol's application messages
  - Avoids mimicking issues
  - Still needs to ensure the cover protocol does not generate "weird" traffic patterns

- Examples:
  - **VoIP/video:** FreeWave [Houmansadr et al. NDSS'13] , DeltaShaper [Barradas et al. PoPETs'17], Protozoa [Barradas et al. CCS'21]
  - **HTTPS:** meek [Fifield et al. PoPETs'15] , decoy routing [Wustrow et al. USENIX Sec'11]
  - **IM/e-mail:** Camoufler [Sharma et al. AsiaCCS'21], SWEET [Houmansadr et al. IEEE/ACM ToN.25]
  - **Cellphones:** Dolphin [Sharma et al. PoPETs' 23]

Issues with traffic tunneling systems

- Oftentimes, there is a disconnect between the usage patterns of the cover protocol and the covert protocol
  - Times of use, duration, etc.
  - The “greedy” tunneling of covert data may change the cover protocol’s typical traffic patterns
    - e.g., exchanging very large IMs very frequently on both directions
  - Covert data embedding mechanisms may slow down the cover’s protocol activity, leading to noticeable changes in traffic patterns
    - e.g., when replacing media data with covert content

Takeaways

- Covert channels allow for the surreptitious transfer of information, both within processes of a given machine or across machines

- Network covert channels are increasingly hard to detect, but can also be used for commendable purposes (e.g., censorship evasion within repressive environments)