Consiglio Nazionale delle Ricerche

SERICS
SECURITY AND RIGHTS IN THE CYBERSPACE

Malware and Information Hiding

(Attack Models and Some Pitfalls)

Luca Caviglione

Institute for Applied Mathematics and Information Technologies

luca.caviglione@cnr.it

University of Pavia - Department of Electrical, Computer and Biomedical Engineering
Pavia, May 14, 2025

Outline

- Motivations
- (a quick introduction to) Covert Channels
- Network Covert Channels
- Local Covert Channels
- Steganographic Malware
- Main Challenges
- Conclusions
- Selected References

Motivations

- Exponential **growth** of malicious software.
- Despite the effort of many security experts and researchers:
  - countermeasures are progressively showing limitations
  - only a fraction of threats is detected
  - malware increasingly operates **undisturbed** for **longer timeframes**.

Malware	Discovered	Present since...
Stuxnet	2010	2007
Duqu	2011	2008
Flame	2012	2007
The Mask	2013	2007
Regin	2014	2003

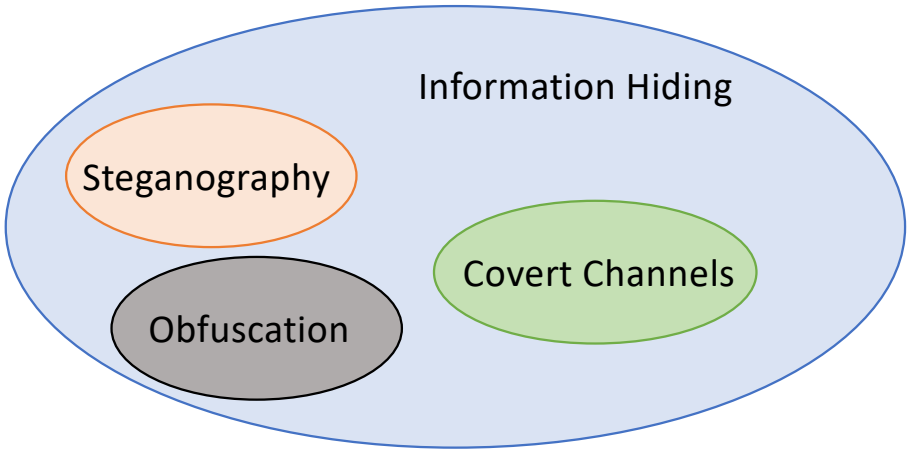
Motivations

- Exponential **growth** of malicious software.
- Despite the effort of many security experts and researchers:
  - countermeasures are progressively showing limitations
  - only a fraction of threats is detected
  - malware increasingly operates **undisturbed** for **longer timeframes**.
- How can malware developers avoid detection for long periods?

Giving an answer is **not** simple!

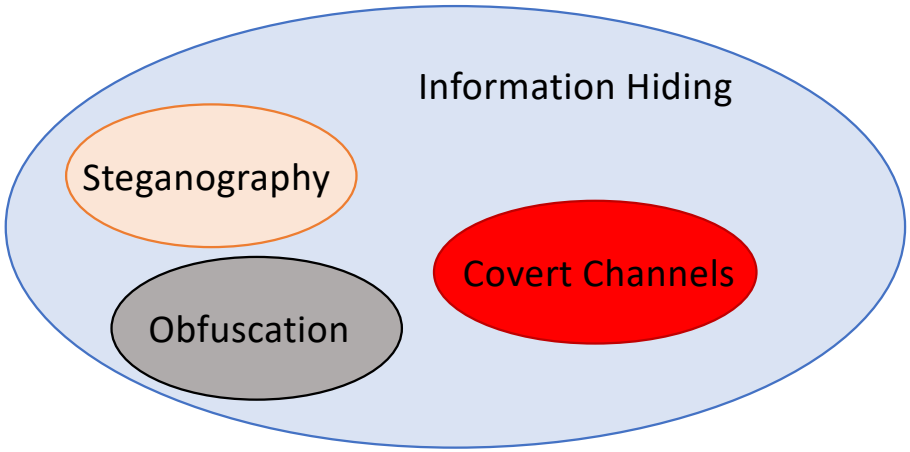
Motivations

- Some possible reasons are:
  - Modular design for customization (e.g., Regin, Flamer, and Weevil)
  - Multistage loading (e.g., Regin, Stuxnet, and Duqu)
  - Cybercrime-as-a-Service models (e.g., Tox)
  - **Information Hiding** techniques (e.g., Platinum APT).



Motivations

- Some possible reasons are:
  - Modular design for customization (e.g., Regin, Flamer, and Weevil)
  - Multistage loading (e.g., Regin, Stuxnet, and Duqu)
  - Cybercrime-as-a-Service models (e.g., Tox)
  - **Information Hiding** techniques (e.g., Platinum APT).

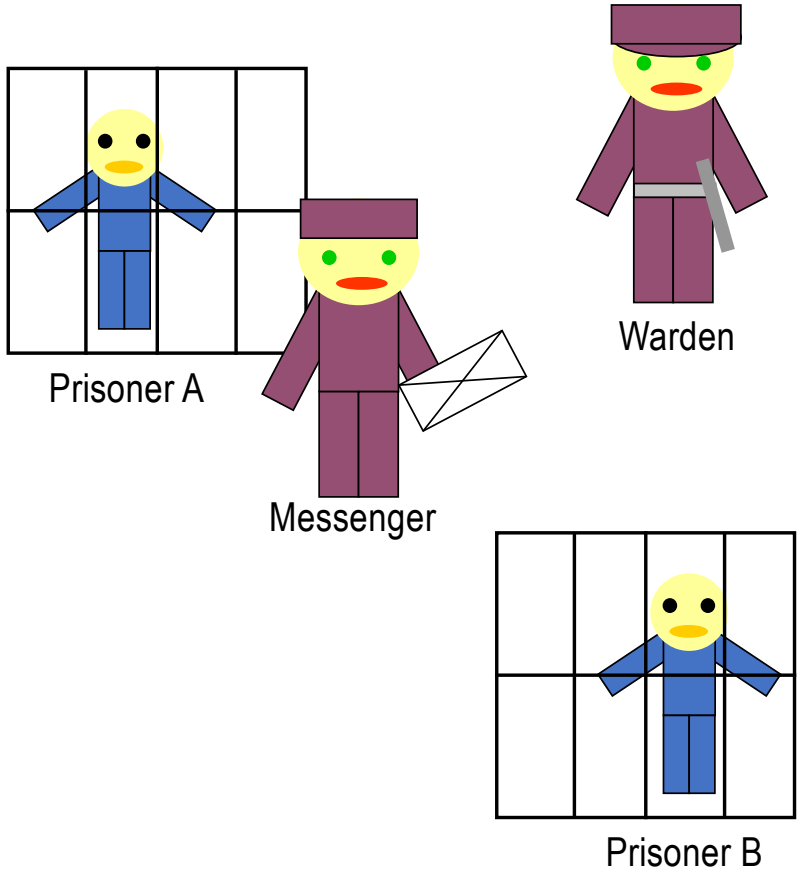


Covert Channels - In Nature

- **Philippine tarsiers** (*Tarsius syrichta*): small nocturnal primates.
- They have a high-frequency auditory sensitivity limit of **91 kHz** and are also able to vocalize with a dominant frequency of **70 kHz**.
- Example of **ultrasonic communication**.
- Philippine tarsiers implement a private **covert communication channel** that cannot be detected by predators, preys, and competitors.

Covert Channels - Models

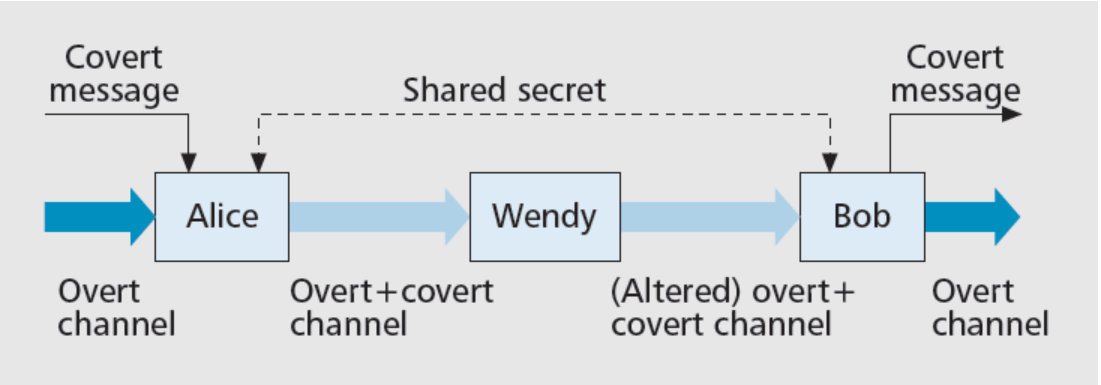
- Modeled by **Simmons** (1984) as the “Prisoners’ Problem and the Subliminal Channel”.



G. J. Simmons, “The Prisoners’ Problem and the Subliminal Channel”, Advances in Cryptology: Proceedings of Crypto 83, pp. 51-67, 1984

Covert Channels - Models

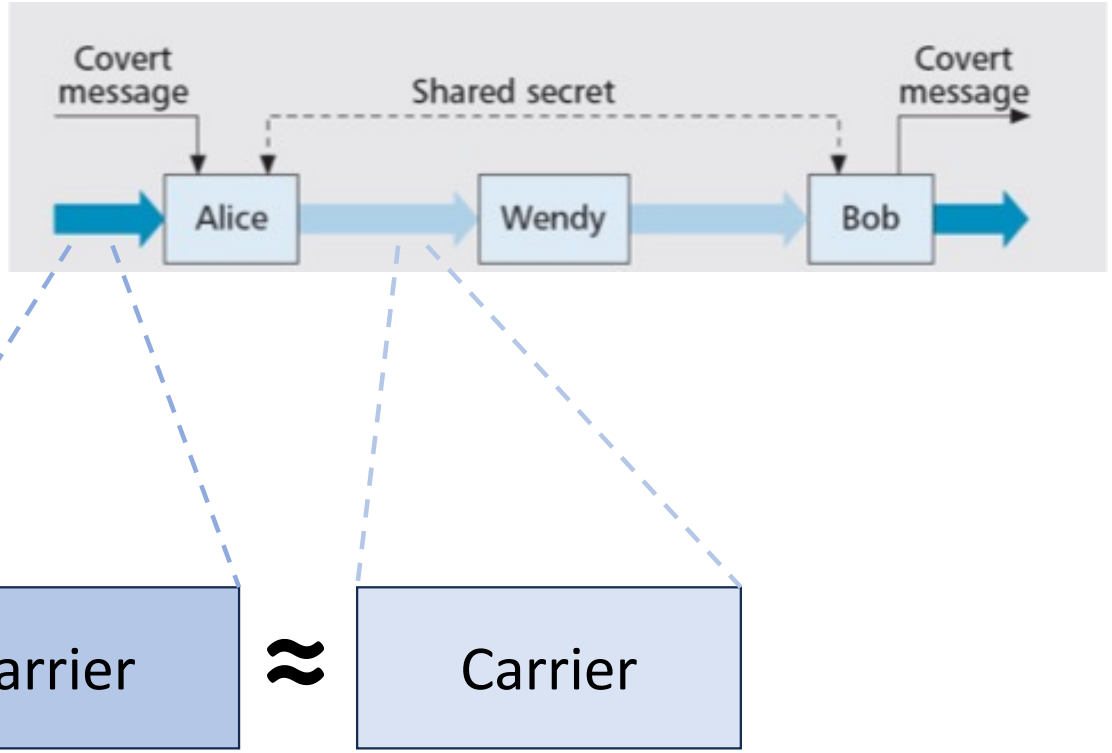
- Modeled by **Simmons** (1984) as the "Prisoners' Problem and the Subliminal Channel".
- According to **Lampson** (1973):
  - *"[channels] not intended for information transfer at all, such as the service program's effect on the system load*".



B. W. Lampson, "A Note on the Confinement Problem", Communications of the ACM, Vol. 16, No. 10, pp. 613-615, Oct. 1973

Covert Channels - Models

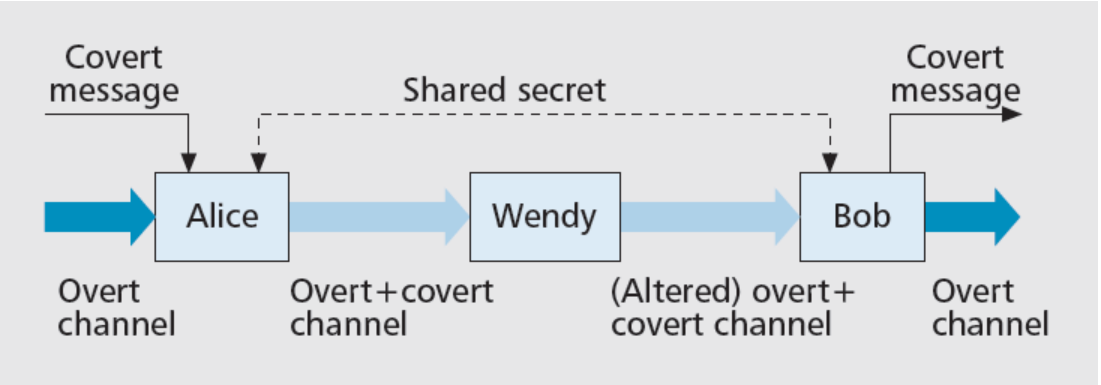
- Modeled by **Simmons** (1984) as the "Prisoners' Problem and the Subliminal Channel".
- According to **Lampson** (1973):
  - *"[channels] not intended for information transfer at all, such as the service program's effect on the system load*".



B. W. Lampson, "A Note on the Confinement Problem", Communications of the ACM, Vol. 16, No. 10, pp. 613-615, Oct. 1973

Covert Channels - Models

- Modeled by **Simmons** (1984) as the “Prisoners’ Problem and the Subliminal Channel”.
- According to **Lampson** (1973):
  - *“[channels] not intended for information transfer at all, such as the service program’s effect on the system load*”.



Information Hiding vs Cryptography

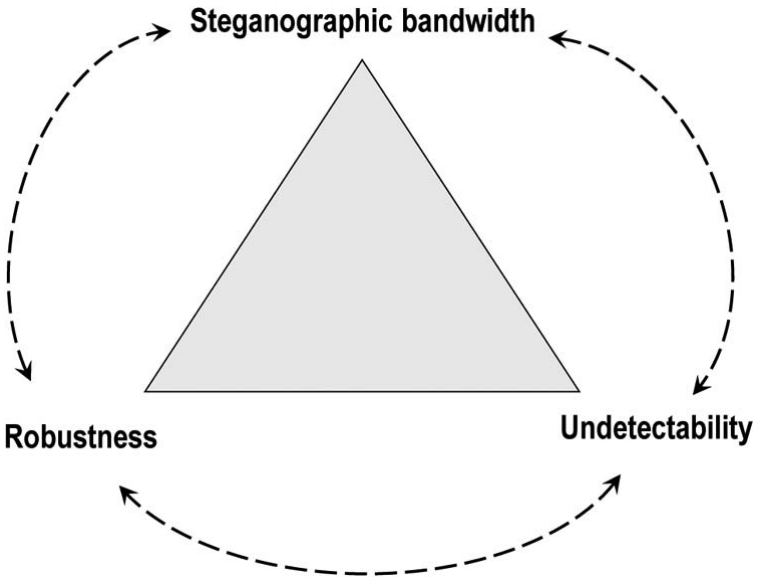
Information Hiding: information is difficult to **notice**
Cryptography: information is difficult to **comprehend**

Covert Channels - Metrics

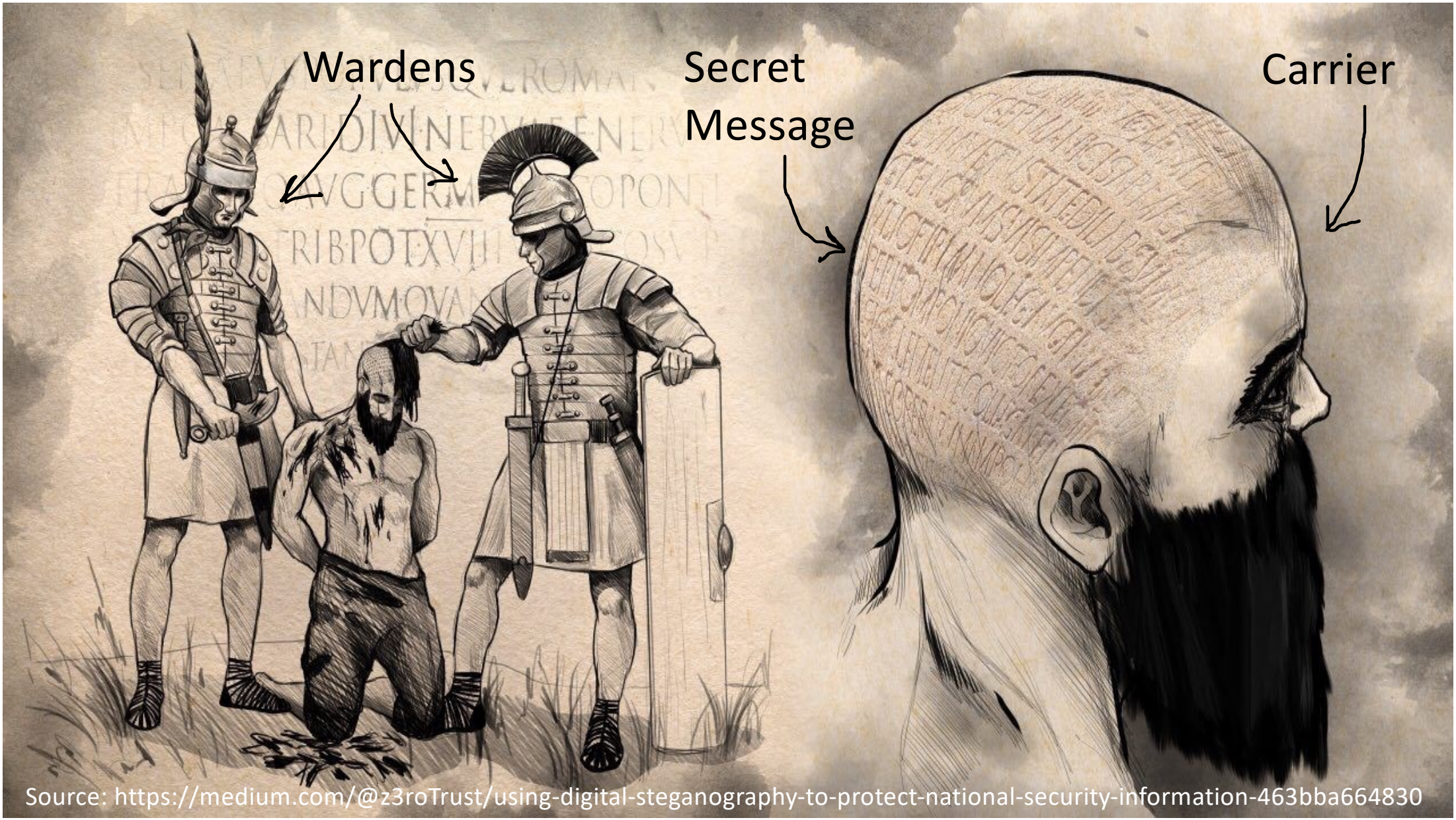
- Covert communications are usually characterized via three metrics:
  - **bandwidth**: the amount of secret data that can be sent per time unit when using a particular method
  - **undetectability**: the inability to detect secret data within a certain carrier
  - **robustness:** the amount of alteration a carrier with covert data can withstand without the secret message being destroyed.

Covert Channels - Metrics

- Covert communications are usually characterized via three metrics:
  - **bandwidth**: the amount of secret data that can be sent per time unit when using a particular method
  - **undetectability**: the inability to detect secret data within a certain carrier
  - **robustness:** the amount of alteration a carrier with covert data can withstand without the secret message being destroyed.
- The metrics are not independent!



Magic Triangle

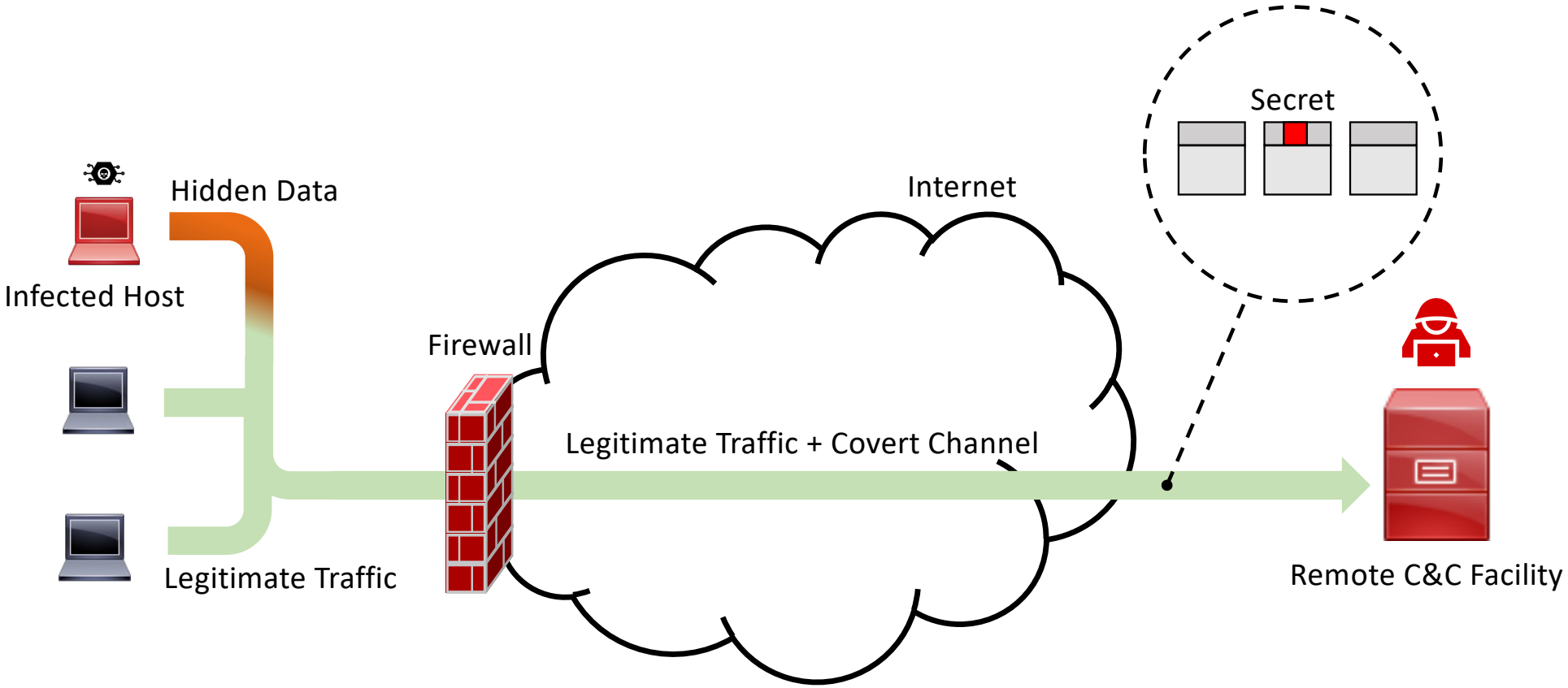


Source: https://medium.com/@z3roTrust/using-digital-steganography-to-protect-national-security-information-463bba664830

Network Covert Channels

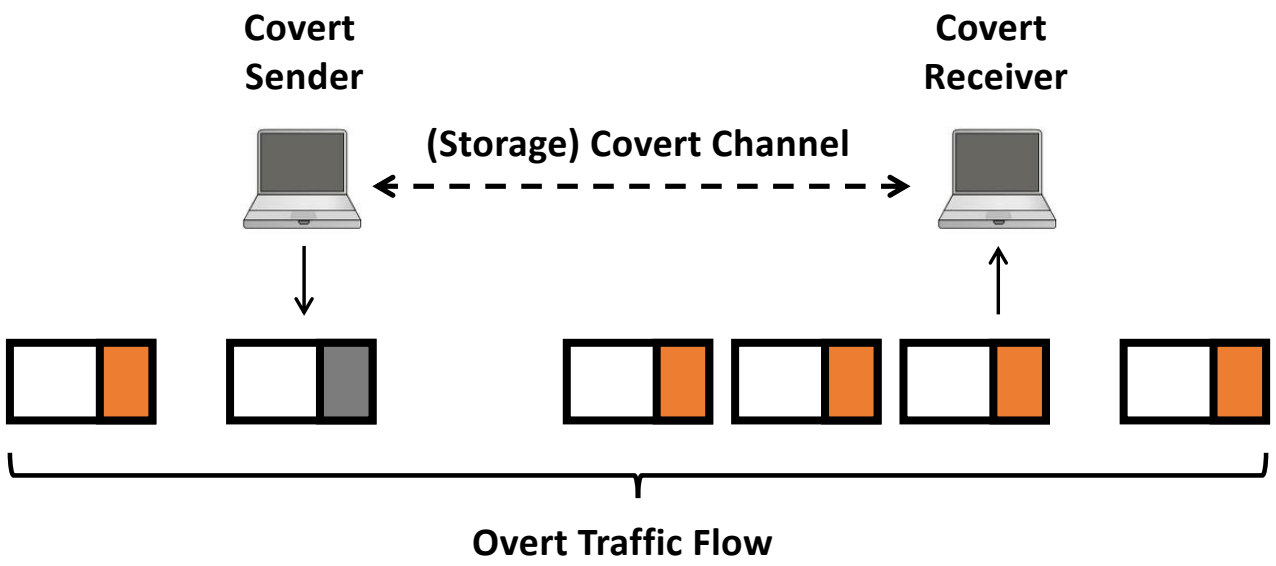
…also denoted as network steganography

Covert Channels in Network Traffic

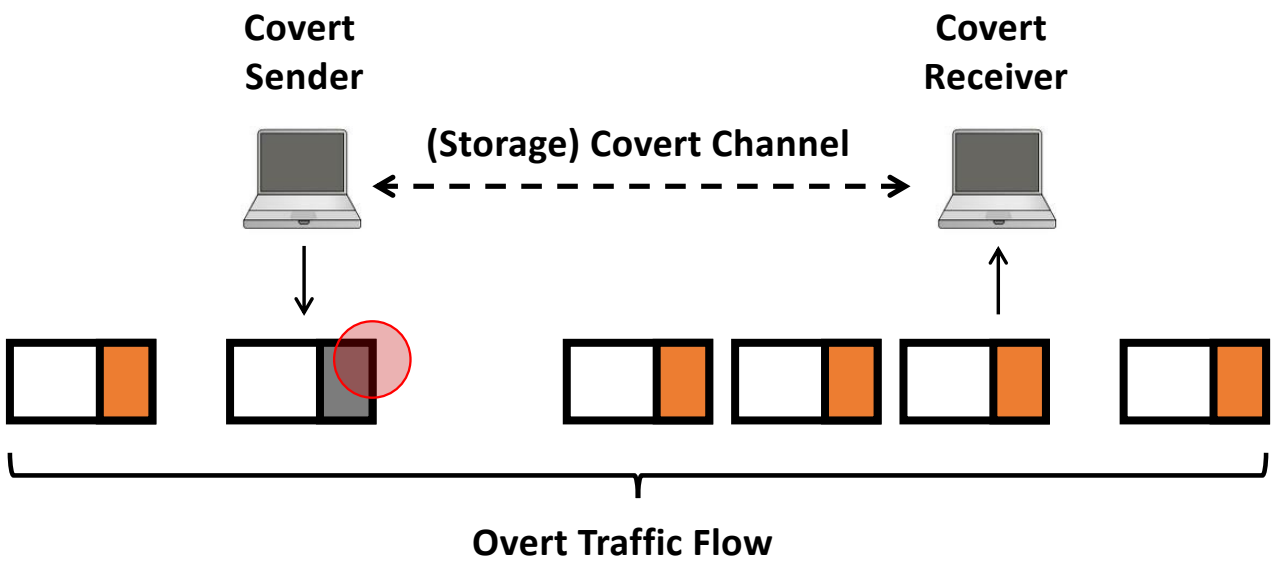


Attack Model

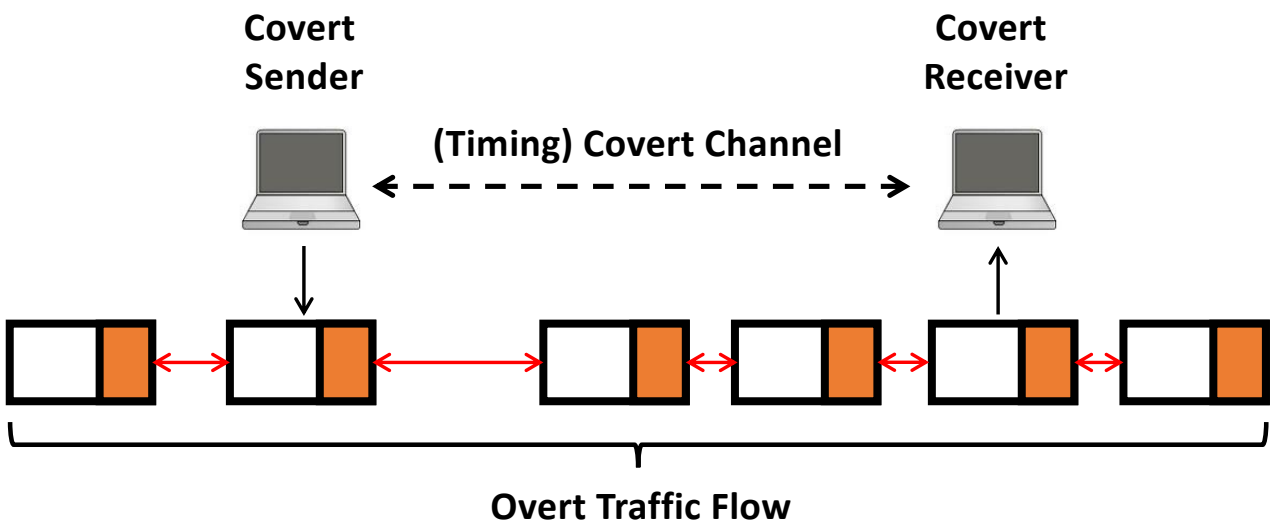
Covert Channels in Network Traffic



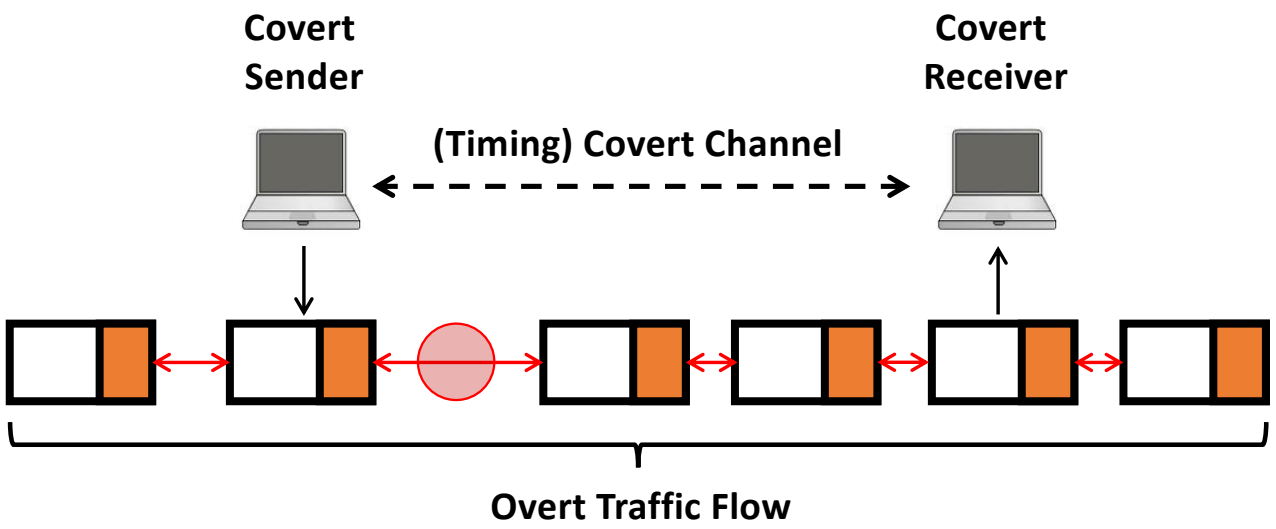
Covert Channels in Network Traffic



Covert Channels in Network Traffic

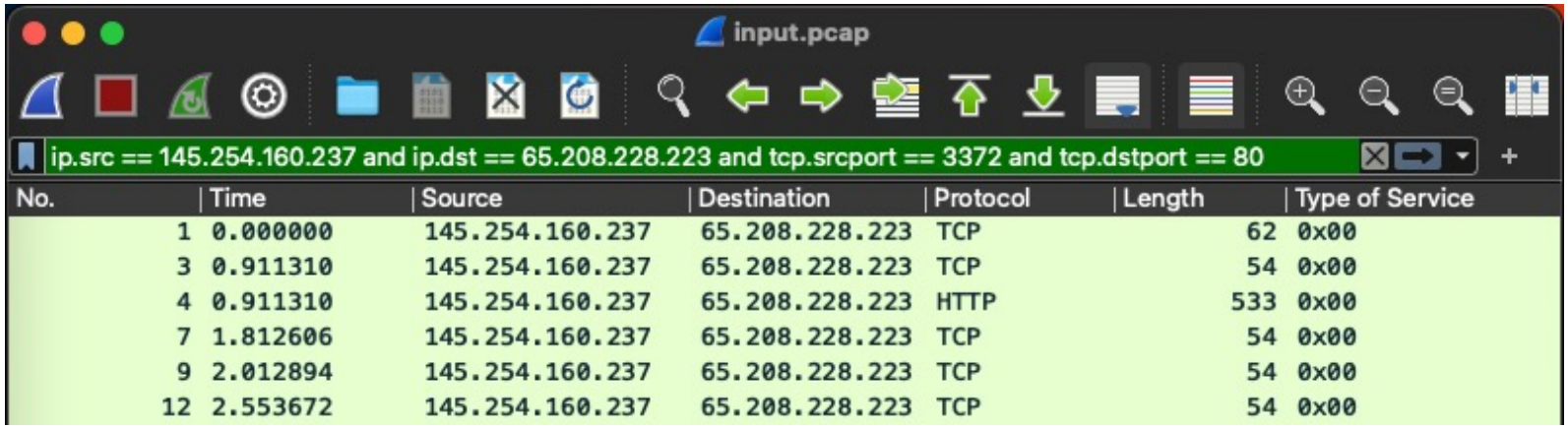


Covert Channels in Network Traffic



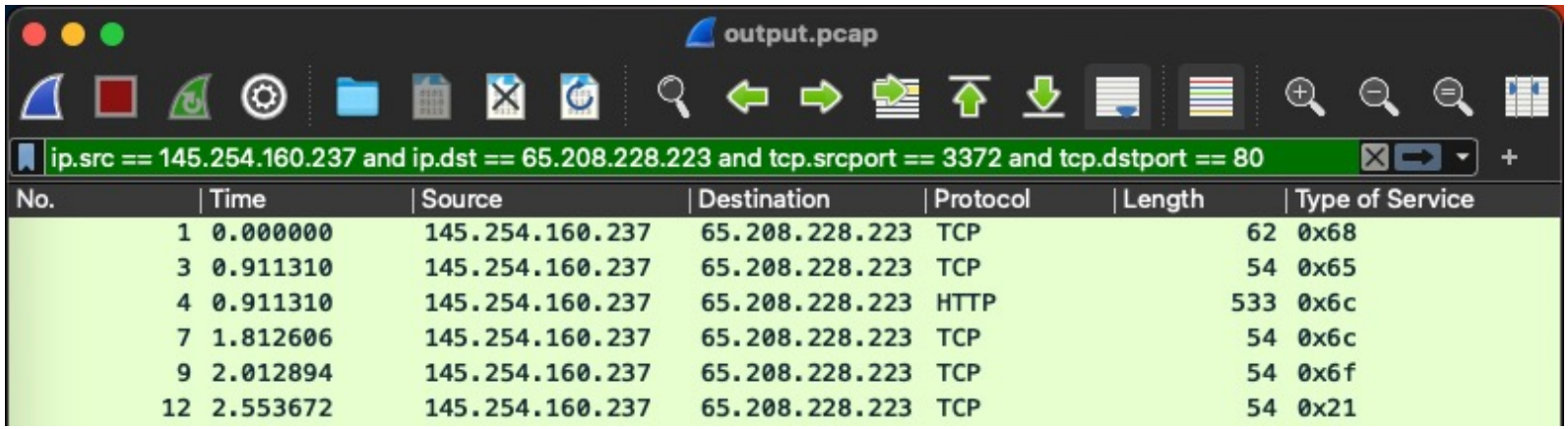
Example - Storage Channel in Type of Service

Overt



No.	Time	Source	Destination	Protocol	Length	Type of Service
1	0.000000	145.254.160.237	65.208.228.223	TCP	62	0x00
3	0.911310	145.254.160.237	65.208.228.223	TCP	54	0x00
4	0.911310	145.254.160.237	65.208.228.223	HTTP	533	0x00
7	1.812606	145.254.160.237	65.208.228.223	TCP	54	0x00
9	2.012894	145.254.160.237	65.208.228.223	TCP	54	0x00
12	2.553672	145.254.160.237	65.208.228.223	TCP	54	0x00

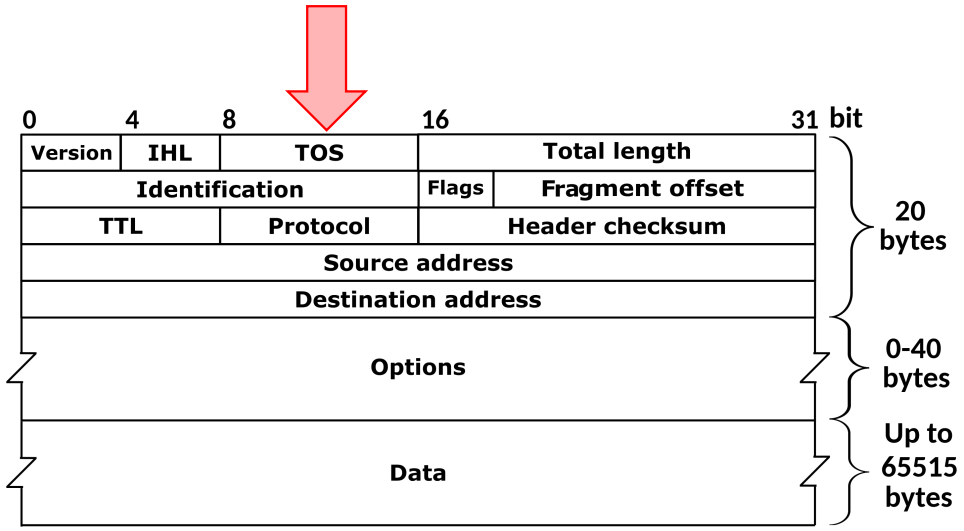
Overt + Covert



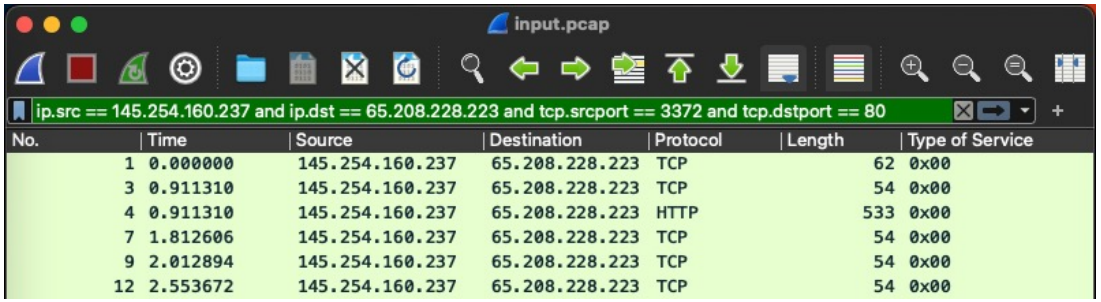
No.	Time	Source	Destination	Protocol	Length	Type of Service
1	0.000000	145.254.160.237	65.208.228.223	TCP	62	0x68
3	0.911310	145.254.160.237	65.208.228.223	TCP	54	0x65
4	0.911310	145.254.160.237	65.208.228.223	HTTP	533	0x6c
7	1.812606	145.254.160.237	65.208.228.223	TCP	54	0x6c
9	2.012894	145.254.160.237	65.208.228.223	TCP	54	0x6f
12	2.553672	145.254.160.237	65.208.228.223	TCP	54	0x21

https://github.com/Ocram95/pcap_injector

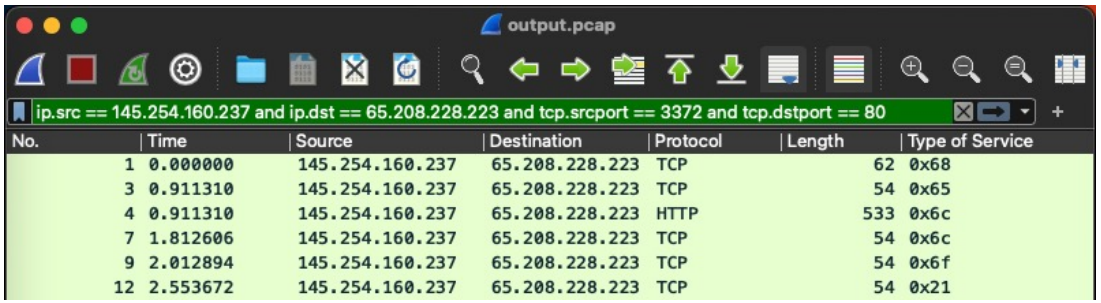
Example - Storage Channel in Type of Service



Overt



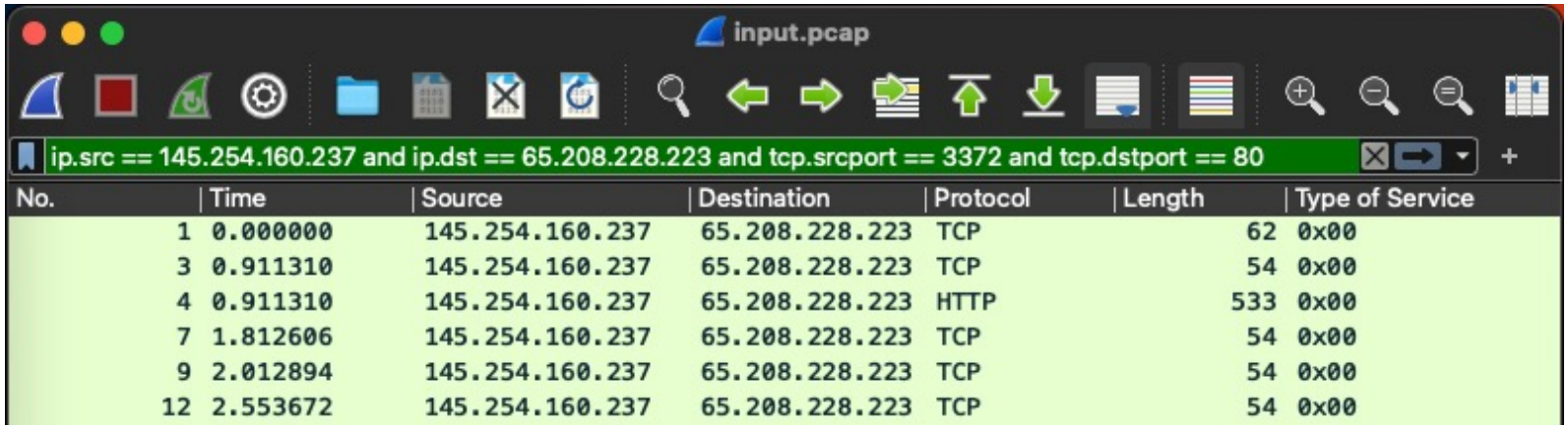
Overt + Covert



https://github.com/Ocram95/pcap_injector

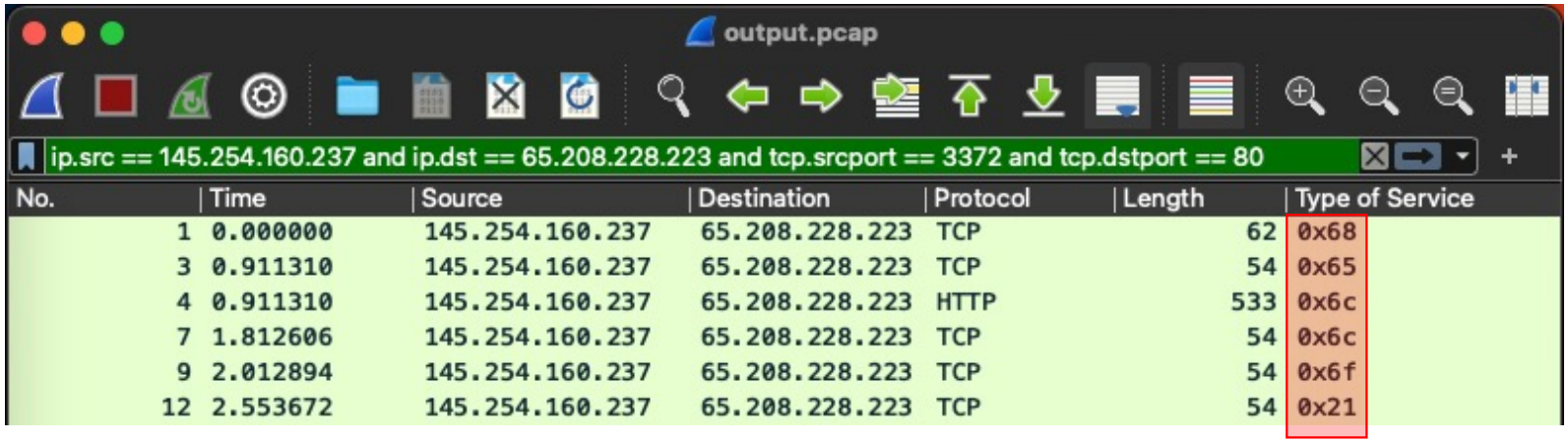
Example - Storage Channel in Type of Service

Overt



No.	Time	Source	Destination	Protocol	Length	Type of Service
1	0.000000	145.254.160.237	65.208.228.223	TCP	62	0x00
3	0.911310	145.254.160.237	65.208.228.223	TCP	54	0x00
4	0.911310	145.254.160.237	65.208.228.223	HTTP	533	0x00
7	1.812606	145.254.160.237	65.208.228.223	TCP	54	0x00
9	2.012894	145.254.160.237	65.208.228.223	TCP	54	0x00
12	2.553672	145.254.160.237	65.208.228.223	TCP	54	0x00

Overt + Covert



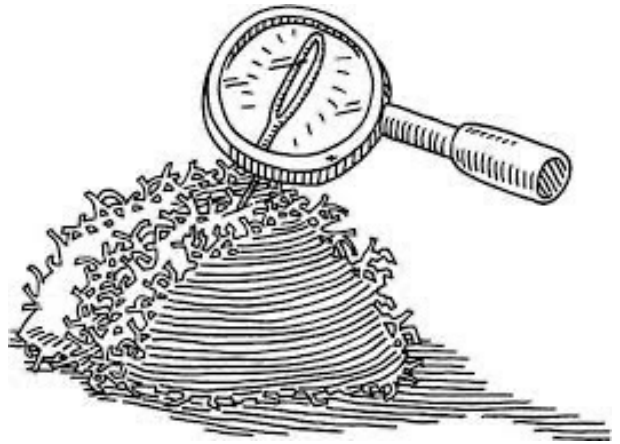
No.	Time	Source	Destination	Protocol	Length	Type of Service
1	0.000000	145.254.160.237	65.208.228.223	TCP	62	0x68
3	0.911310	145.254.160.237	65.208.228.223	TCP	54	0x65
4	0.911310	145.254.160.237	65.208.228.223	HTTP	533	0x6c
7	1.812606	145.254.160.237	65.208.228.223	TCP	54	0x6c
9	2.012894	145.254.160.237	65.208.228.223	TCP	54	0x6f
12	2.553672	145.254.160.237	65.208.228.223	TCP	54	0x21

ASCII for “hello!”

https://github.com/Ocram95/pcap_injector

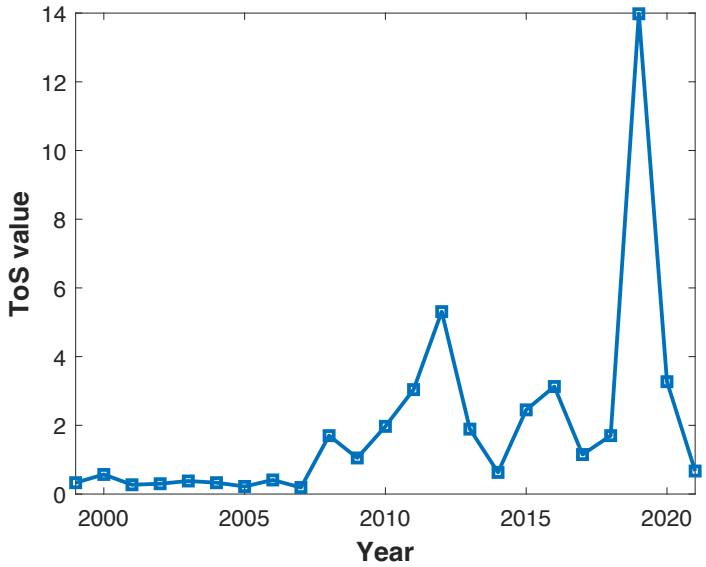
Example - Storage Channel in Type of Service

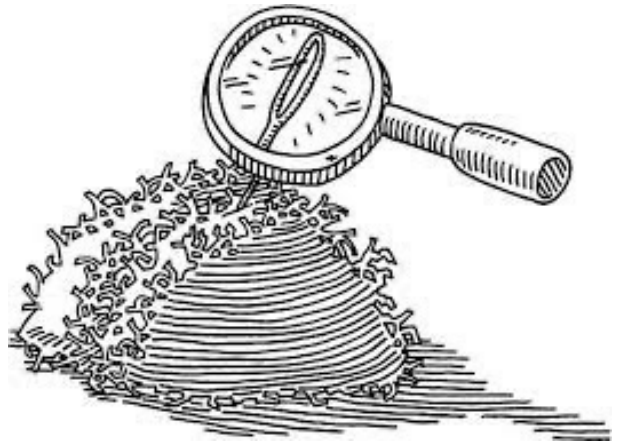
- To be effective, a (network) covert channel should:
  - use a **popular** carrier
  - not reveal the presence of the channel via **anomalous** behaviors
  - **not disrupt** the functionality of a protocol or a service.



Example - Storage Channel in Type of Service

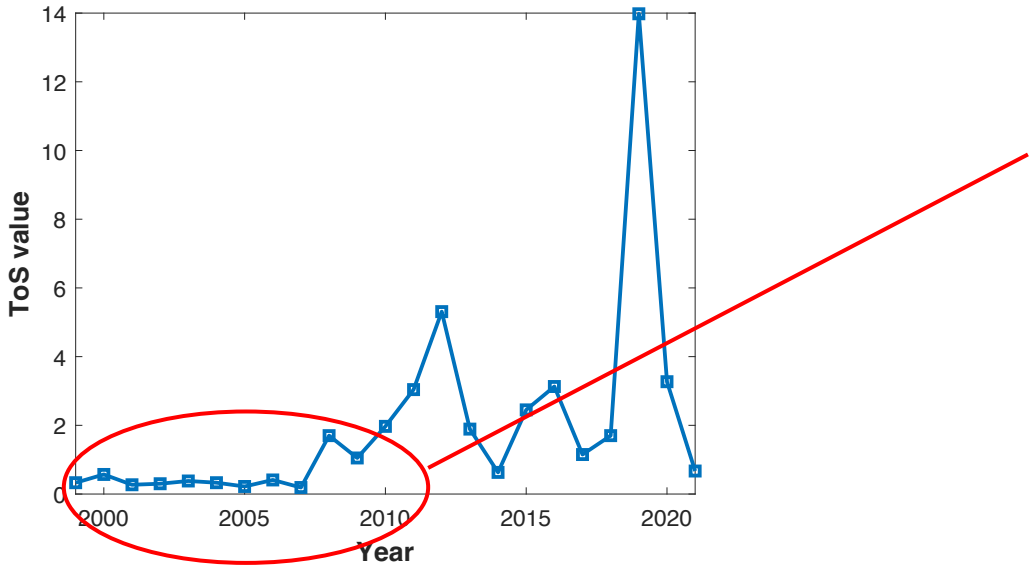
- To be effective, a (network) covert channel should:
  - use a **popular** carrier
  - not reveal the presence of the channel via **anomalous** behaviors
  - **<u>not disrupt</u>** the functionality of a protocol or a service.





Example - Storage Channel in Type of Service

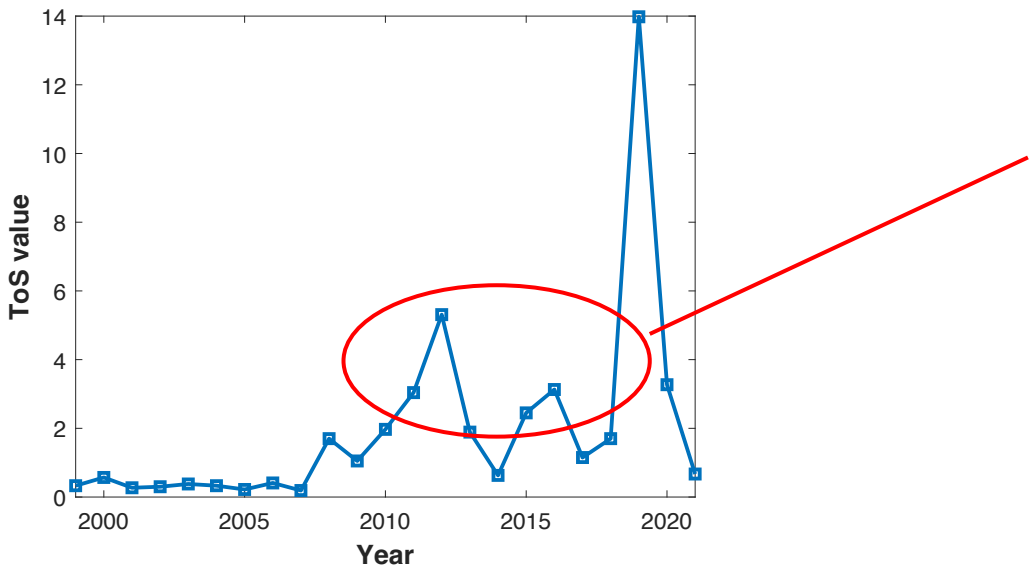
- To be effective, a (network) covert channel should:
  - use a **popular** carrier
  - not reveal the presence of the channel via **anomalous** behaviors
  - **<u>not disrupt</u>** the functionality of a protocol or a service.



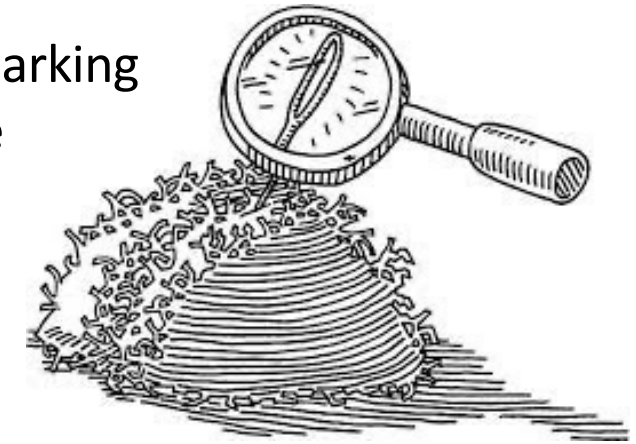
Only few values, thus making the presence of arbitrary data easily detectable.

Example - Storage Channel in Type of Service

- To be effective, a (network) covert channel should:
  - use a **popular** carrier
  - not reveal the presence of the channel via **anomalous** behaviors
  - **<u>not disrupt</u>** the functionality of a protocol or a service.



Pathologies in the modification of DSCP by routers, such as bleaching or remarking could disrupt the channel even if undetected.



A. Custura, R. Secchi, G. Fairhurst. “Exploring DSCP Modification Pathologies in the Internet”, Computer Communications, Vol. 127, pp. 86 - 94, 2018

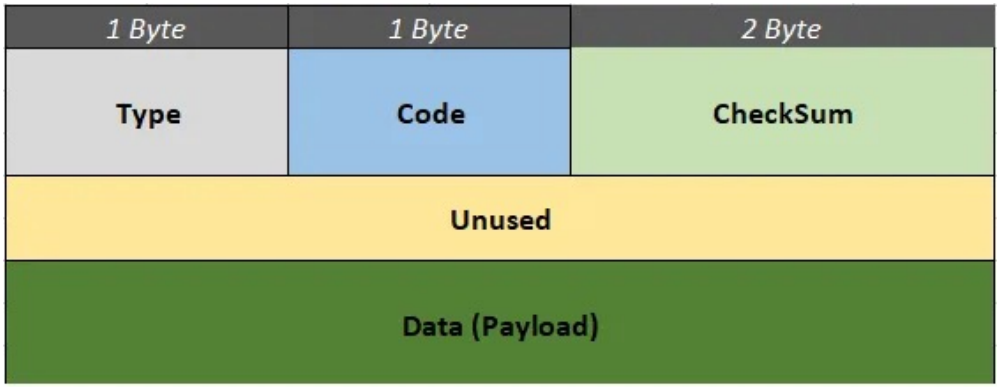
Malware Examples - PlugX 2.0

- Discovered in 2014 (update of the PlugX RAT).
- It uses a custom algorithm for encryption and hides C&C communications in ICMP, DNS, and HTTP protocols.
- Attack Phases:
  - ICMP is used to join the C&C server

Malware Examples - PlugX 2.0

- Discovered in 2014 (update of the PlugX RAT).
- It uses a custom algorithm for encryption and hides C&C communications in ICMP, DNS, and HTTP protocols.
- Attack Phases:
  - ICMP is used to join the C&C server
  - data is transmitted as a payload of `Echo Reply` packets (ICMP Type 0)

Malicious Data



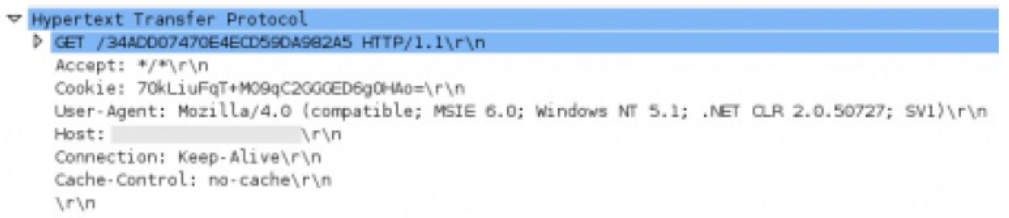
Malware Examples - PlugX 2.0

- Discovered in 2014 (update of the PlugX RAT).
- It uses a custom algorithm for encryption and hides C&C communications in ICMP, DNS, and HTTP protocols.
- Attack Phases:
  - ICMP is used to join the C&C server
  - data is transmitted as a payload of `Echo Reply` packets (ICMP Type 0)
  - HTTP to transport signaling

POST/%p%p%p

Method 1

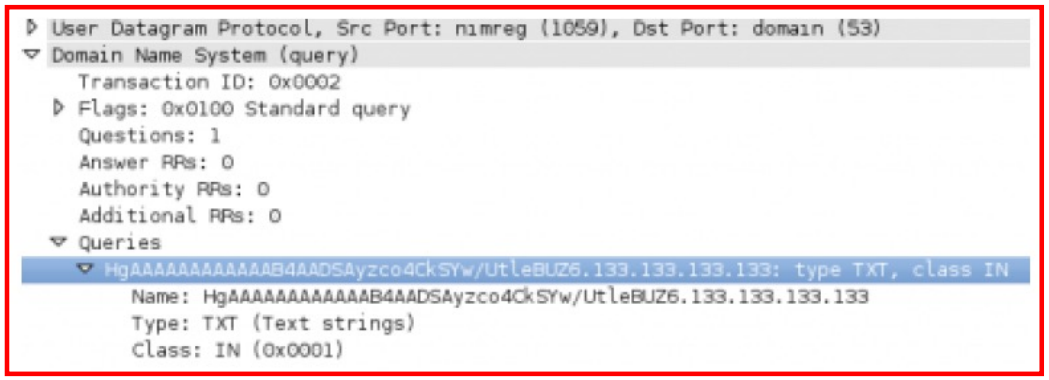
GET (data base64-encoded in Cookie header)



Method 2

Malware Examples - PlugX 2.0

- Discovered in 2014 (update of the PlugX RAT).
- It uses a custom algorithm for encryption and hides C&C communications in ICMP, DNS, and HTTP protocols.
- Attack Phases:
  - ICMP is used to join the C&C server
  - data is transmitted as a payload of `Echo Reply` packets (ICMP Type 0)
  - HTTP to transport signaling
  - data is sent in DNS traffic.

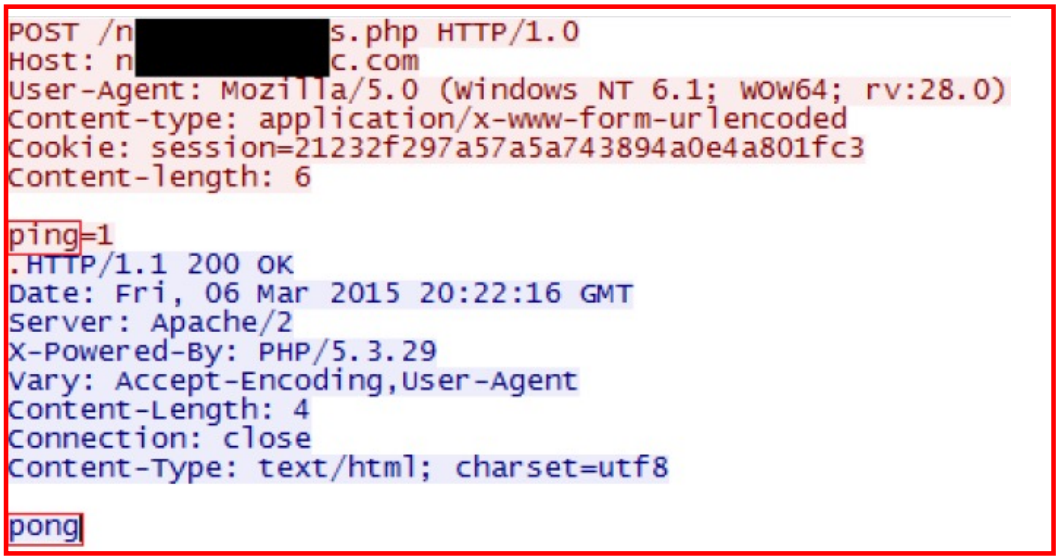


Malware Examples - W32/Foreign.LXES!tr

- Discovered in April 2015.
- It hides malicious traffic within HTTP communications.
- Attack Phases:
  - the C&C server status is checked with ping-pongs HTTP `POST/200 OK` messages

Malware Examples - W32/Foreign.LXES!tr

- Discovered in April 2015.
- It hides malicious traffic within HTTP communications.
- Attack Phases:
  - the C&C server status is checked with ping-pongs HTTP `POST/200 OK` messages



Malware Examples - W32/Foreign.LXES!tr

- Discovered in April 2015.
- It hides malicious traffic within HTTP communications.
- Attack Phases:
  - the C&C server status is checked with ping-pongs HTTP POST/200 OK messages
  - data is hidden in the source code of 404 messages.

Detection and Mitigation

- Detection and mitigation techniques against network covert channels can be broadly classified in three groups:
  - **normalization**: removing ambiguities in traffic that can be used as a carrier
  - **statistical methods**: recognize traits that can be used to hide data
  - **AI**: trained classifiers capable of distinguishing between traffic with a covert channel and normal network flow.

Detection and Mitigation

- Detection and mitigation techniques against network covert channels can be broadly classified in three groups:
  - **normalization**: removing ambiguities in traffic that can be used as a carrier
  - **statistical methods**: recognize traits that can be used to hide data
  - **AI**: trained classifiers capable of distinguishing between traffic with a covert channel and normal network flow.

Penalizes everything and everyone.

Detection and Mitigation

- Detection and mitigation techniques against network covert channels can be broadly classified in three groups:
  - **normalization**: removing ambiguities in traffic that can be used as a carrier
  - **statistical methods**: recognize traits that can be used to hide data
  - **AI**: trained classifiers capable of distinguishing between traffic with a covert channel and normal network flow.

Requires modeling traffic or the “normal” behavior of the workload.

Detection and Mitigation

- Detection and mitigation techniques against network covert channels can be broadly classified in three groups:
  - **normalization**: removing ambiguities in traffic that can be used as a carrier
  - **statistical methods**: recognize traits that can be used to hide data
  - **AI**: trained classifiers capable of distinguishing between traffic with a covert channel and normal network flow.

Preparing datasets is not easy, AI should be explainable, and we cannot inspect what we want.

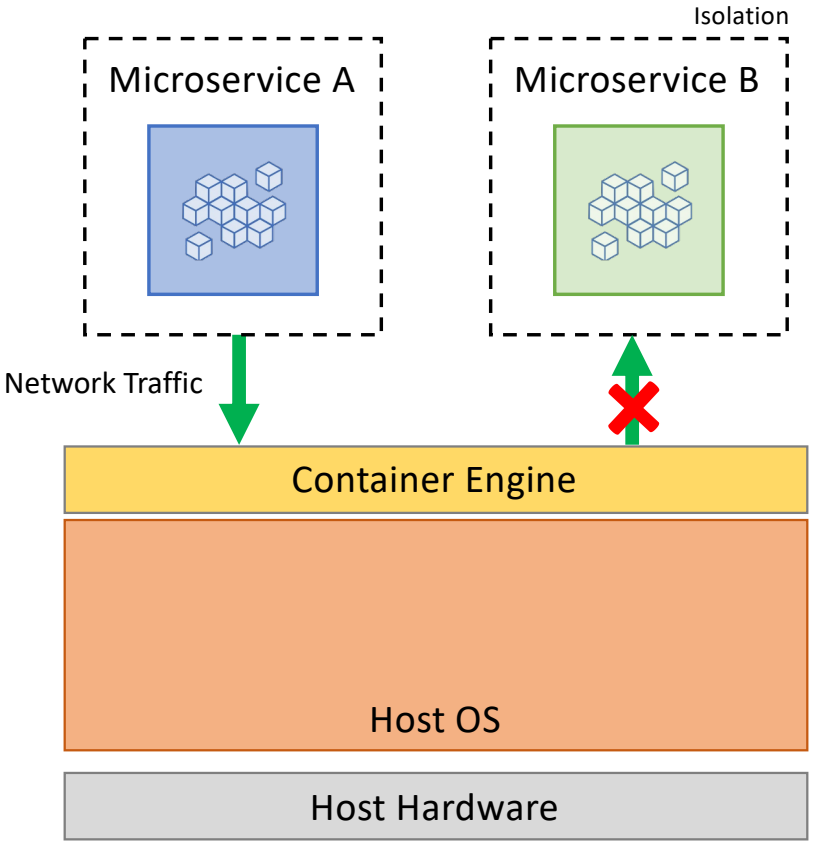
Detection and Mitigation

- Detection and mitigation techniques against network covert channels can be broadly classified in three groups:
  - **normalization**: removing ambiguities in traffic that can be used as a carrier
  - **statistical methods**: recognize traits that can be used to hide data
  - **AI**: trained classifiers capable of distinguishing between traffic with a covert channel and normal network flow.
- Very difficult to do *a posteriori*.

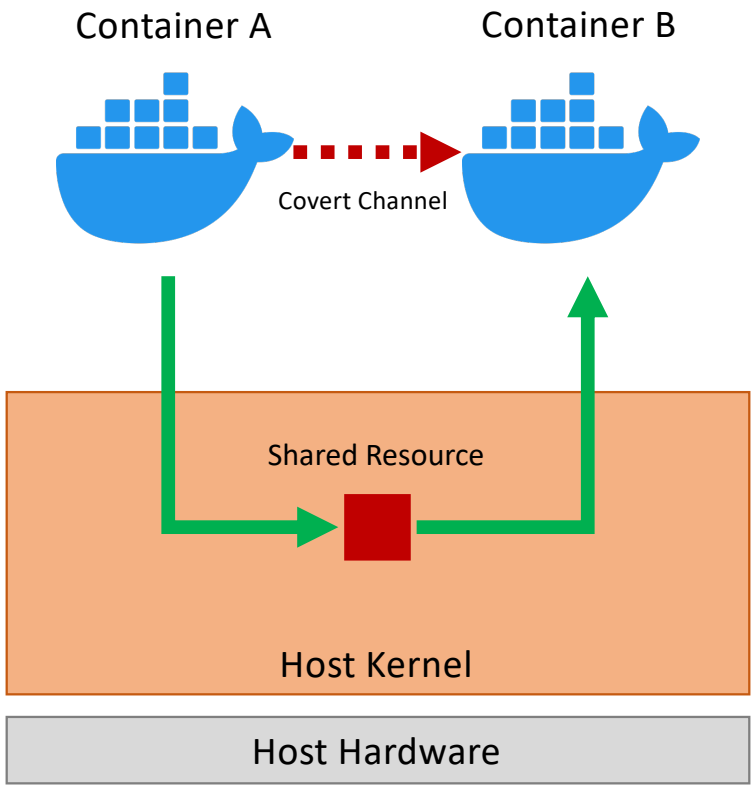
Local Covert Channels

…including some ideas for pursuing abstraction

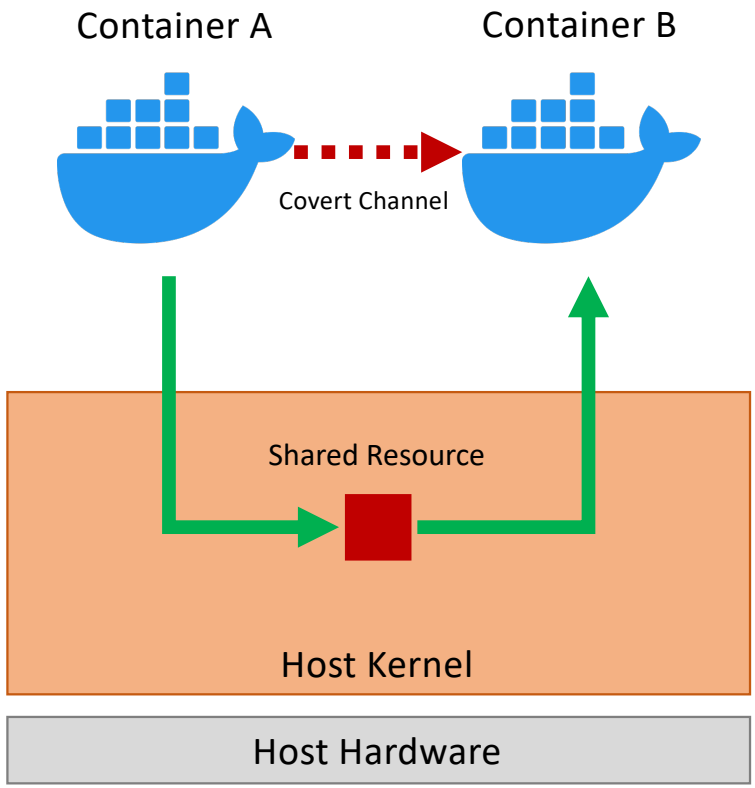
Covert Channels and Microservices



Covert Channels in Docker Containers

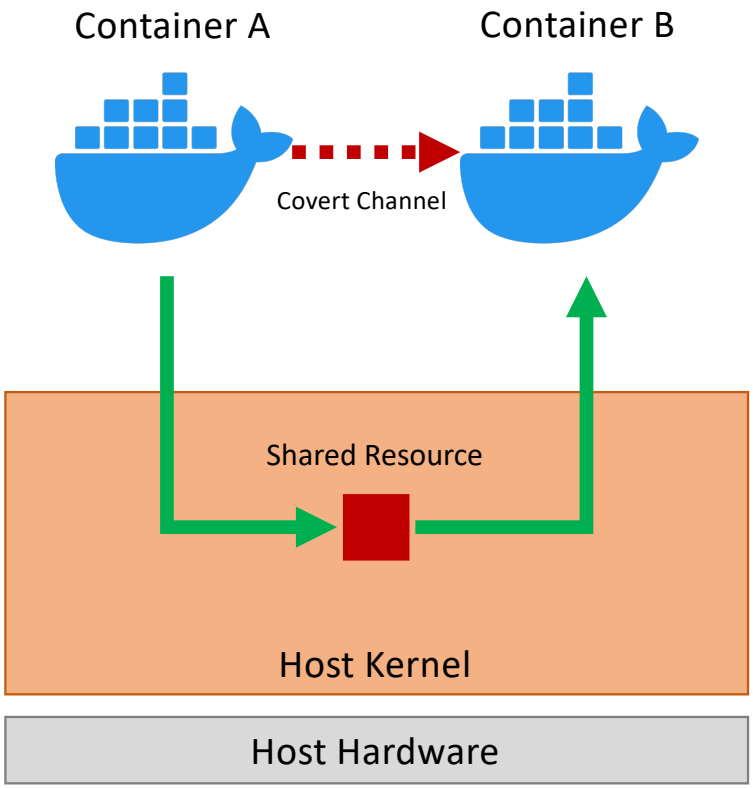


Covert Channels in Docker Containers



- Containers can be targeted with a covert channel to:
  - leak information
  - orchestrate attacks
  - evaluate co-residence (reconnaissance).
- The carrier could be a loosely-isolated shared resource.
- Effective approaches rely on the manipulations of **software** and **hardware statistics** accessible through the `/proc` filesystem:
  - CPU load
  - threads enumeration
  - memory patterns
  - …

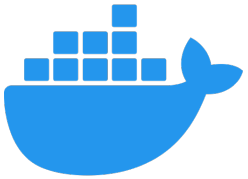
Covert Channels in Docker Containers

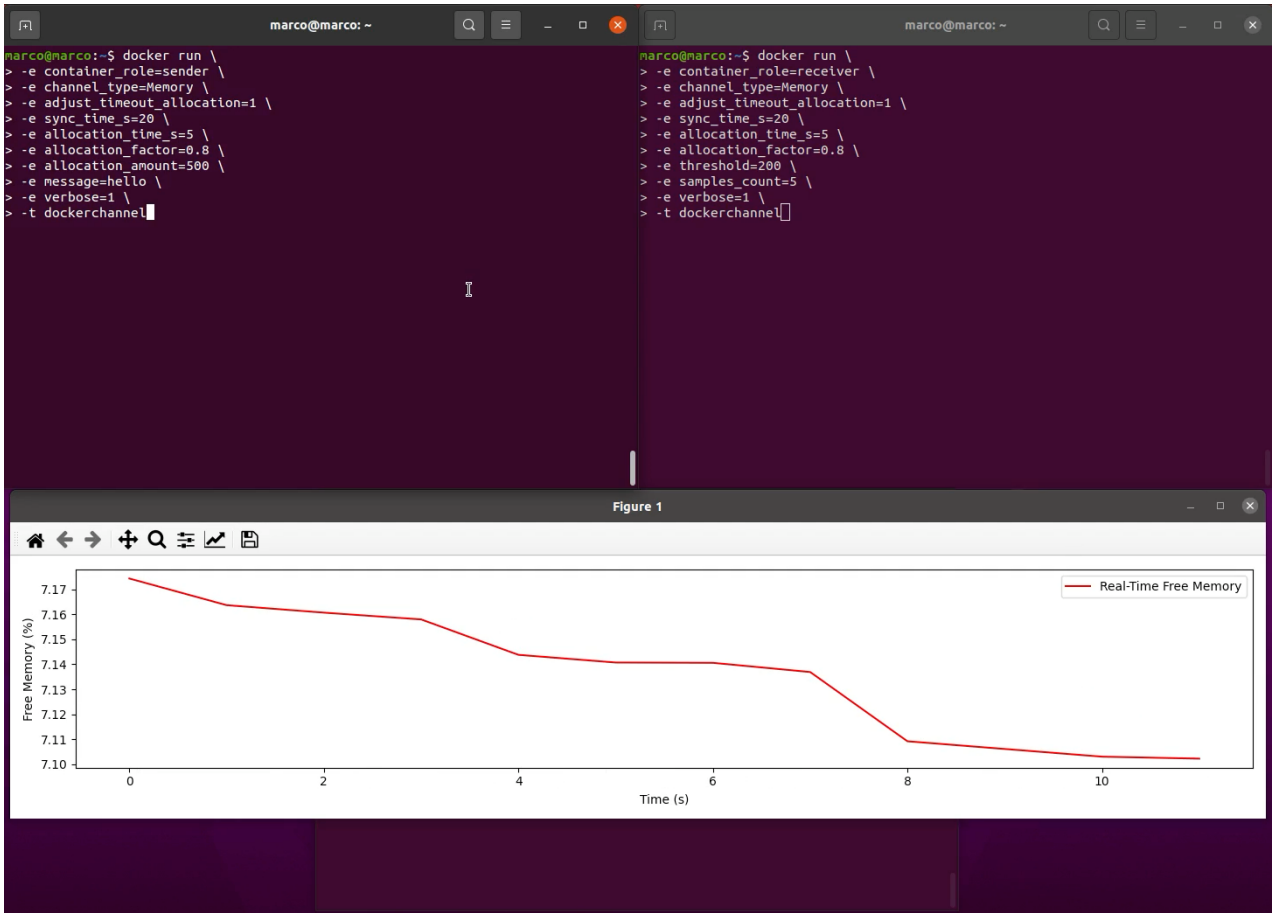


- Example with containers exploiting the free memory available of the host and visible via `/proc/meminfo`.
- **Container A** encodes a binary message by altering the amount of used memory:
  - it allocates 500 MB of memory to encode the bit 1 and sleeps for 5 seconds
  - it sleeps for 5 seconds to encode the bit 0.
- **Container B** periodically monitors the free memory:
  - the bit 1 is decoded if the difference between the average current memory usage and the previous one exceeds by a threshold of 200 MB
  - otherwise, the bit 0 is decoded.

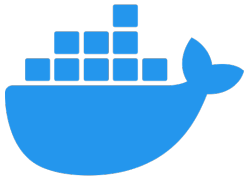
Covert Channels in Docker Containers

Container A





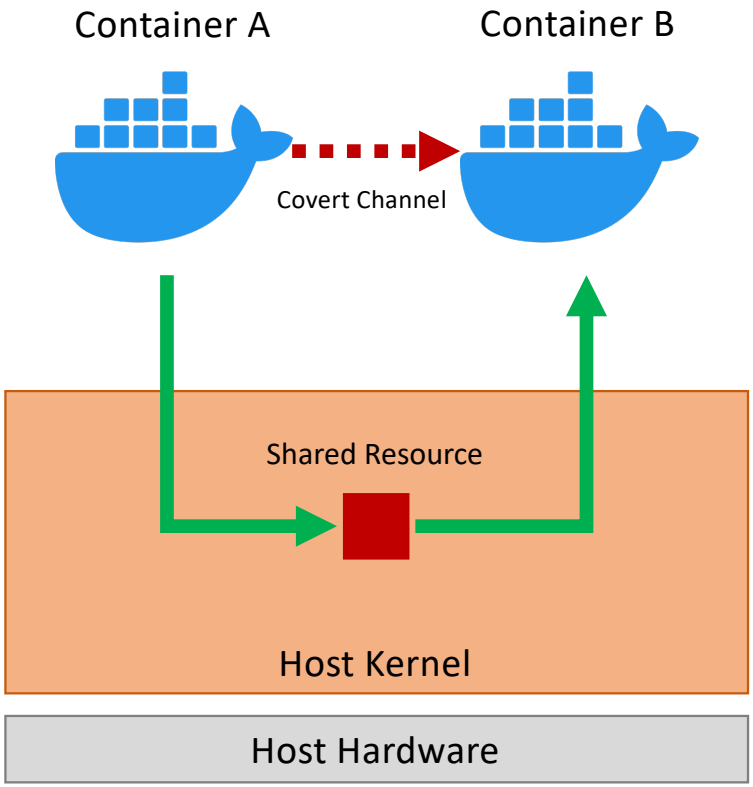
Container B



Shared Resource

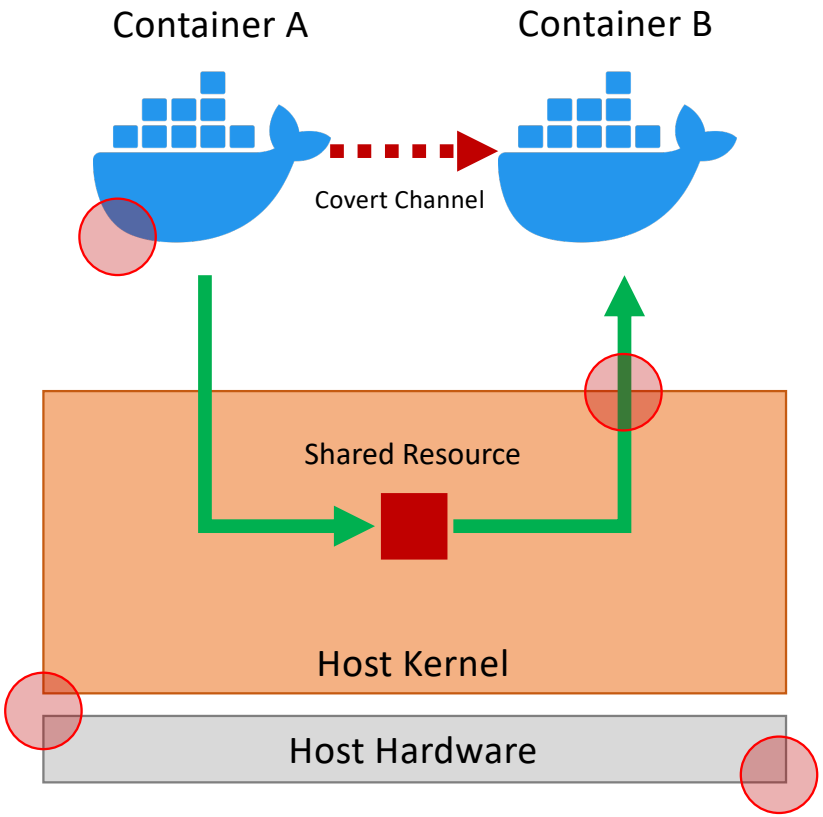
https://github.com/cybersecurity-cnr/DockerChannel

Covert Channels in Docker Containers



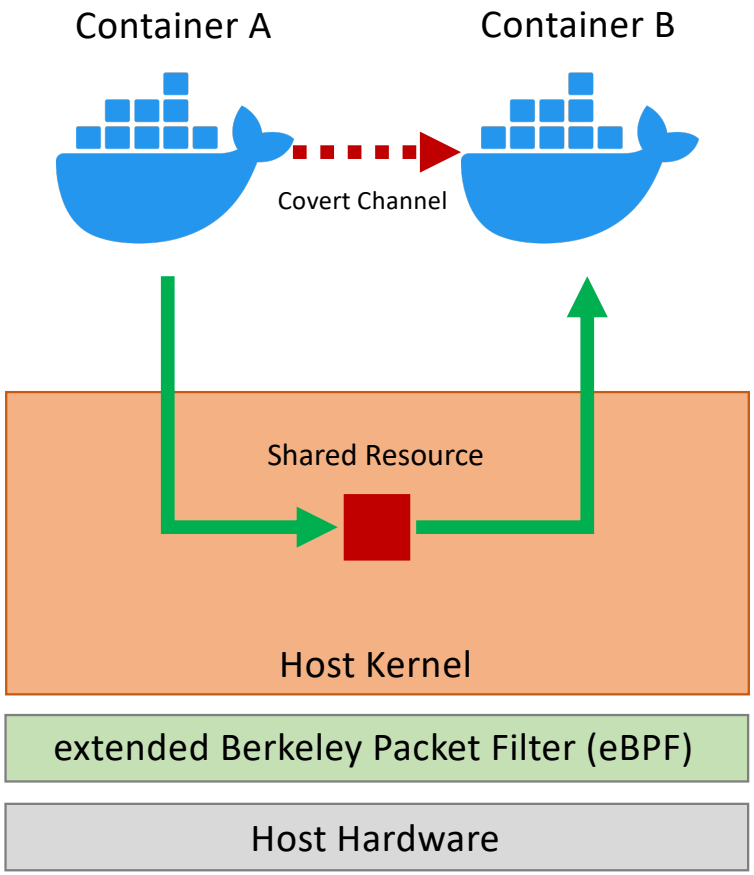
- A possible detection approach is to identify suitable patterns or specific behaviors characterizing the leakage.

Covert Channels in Docker Containers



- A possible detection approach is to identify suitable patterns or specific behaviors characterizing the leakage.
- **Problem 1**: where to perform inspection.

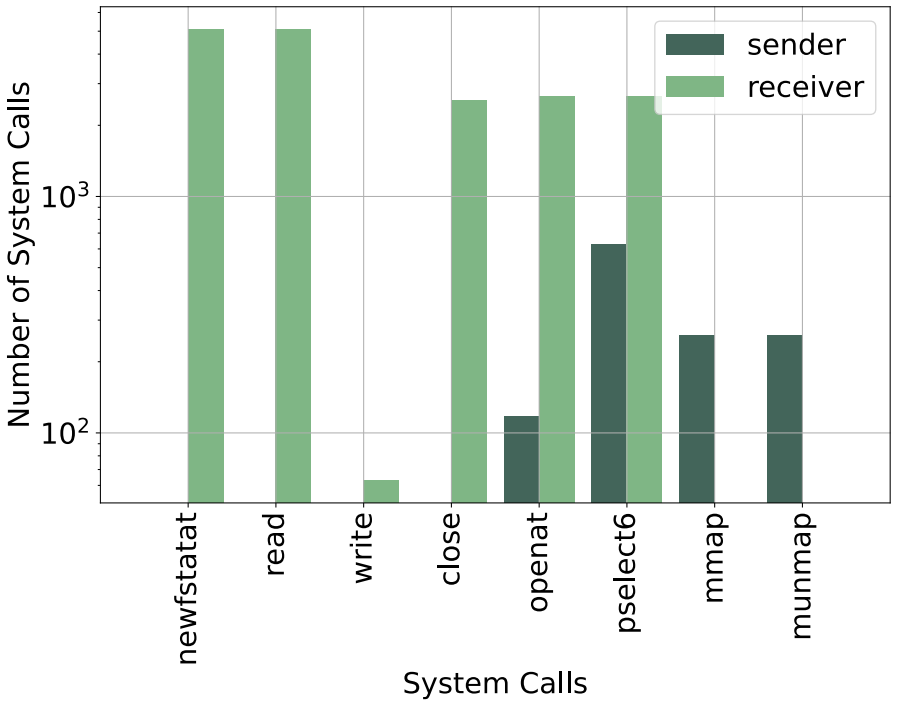
Covert Channels in Docker Containers



- A possible detection approach is to identify suitable patterns or specific behaviors characterizing the leakage.
- Problem 1: where to perform inspection.
- **Problem 2:** how to perform inspection.

https://ebpf.io

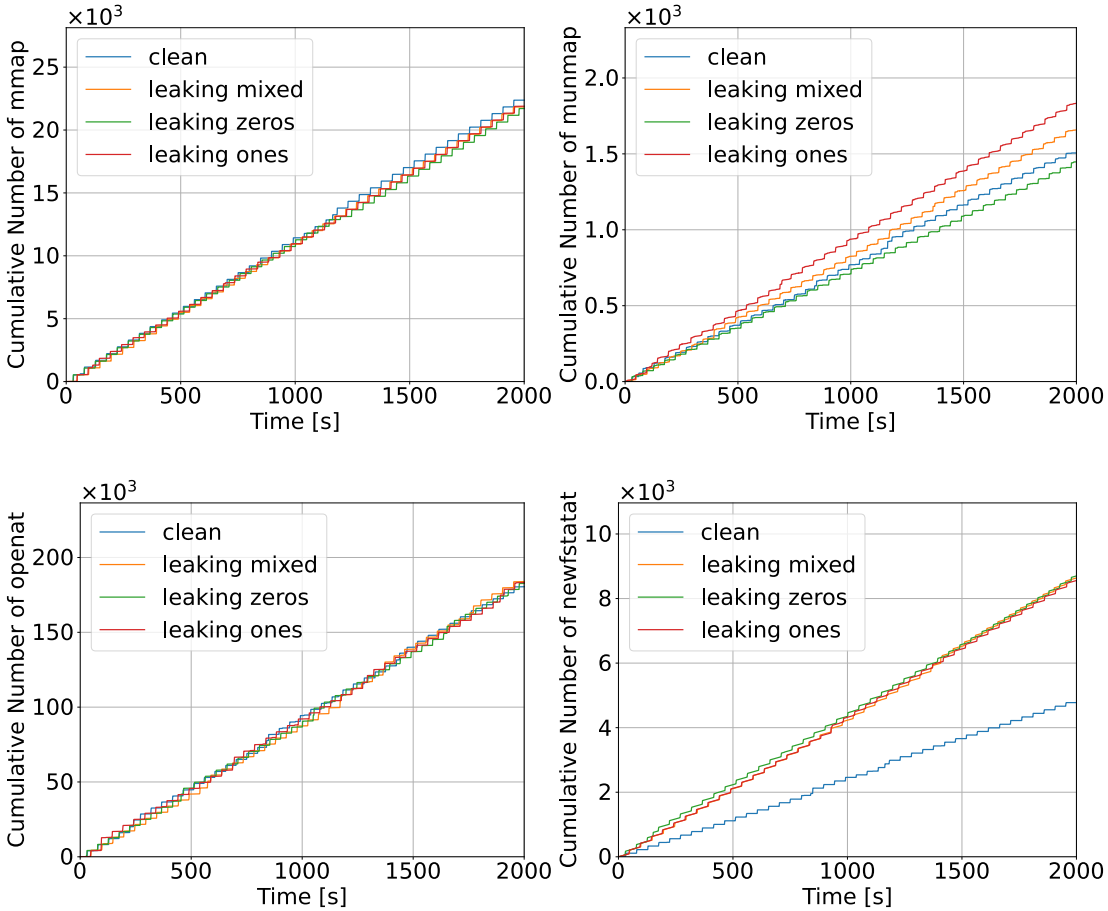
Covert Channels in Docker Containers



- A possible detection approach is to identify suitable patterns or specific behaviors characterizing the leakage.
- Problem 1: where to perform inspection.
- Problem 2: how to perform inspection.
- **Problem 3**: what to inspect.

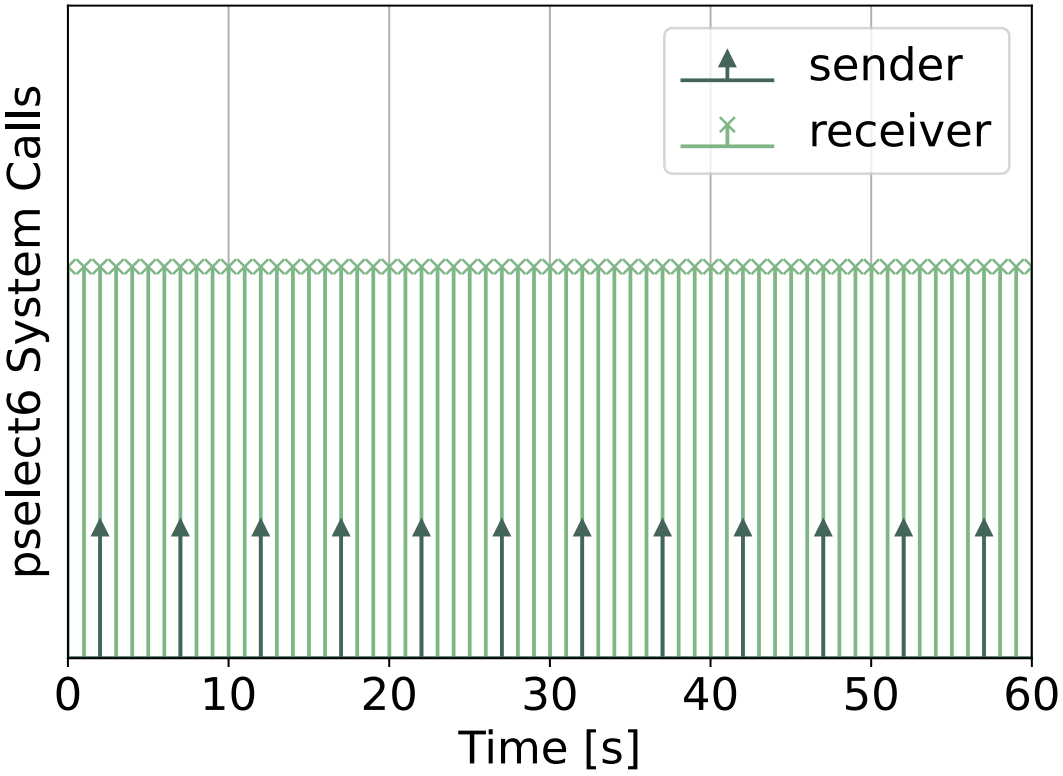
Covert Channels in Docker Containers

- **Example**: a “clean” container compared to another container leaking different messages (512 bits):
  - “leaking mixed”: alternating 0 and 1 bits
  - “leaking zeros”: only 0 bits
  - “leaking ones”: only 1 bits
- Results show the `mmap`, the `munmap`, the `openat`, and the `newfstatat` system calls.



Abstraction Example - Sleeping

- To avoid being too carrier-dependent, pursuing **abstraction** is desirable.
- **Fact**: the sender and the receiver should act close in time to prevent other processes/containers disrupting the channel.
- **Idea**: spot tight time correlations among containers.
- The `pselect6` is a good **indicator** as it is used to put processes in a sleep state.

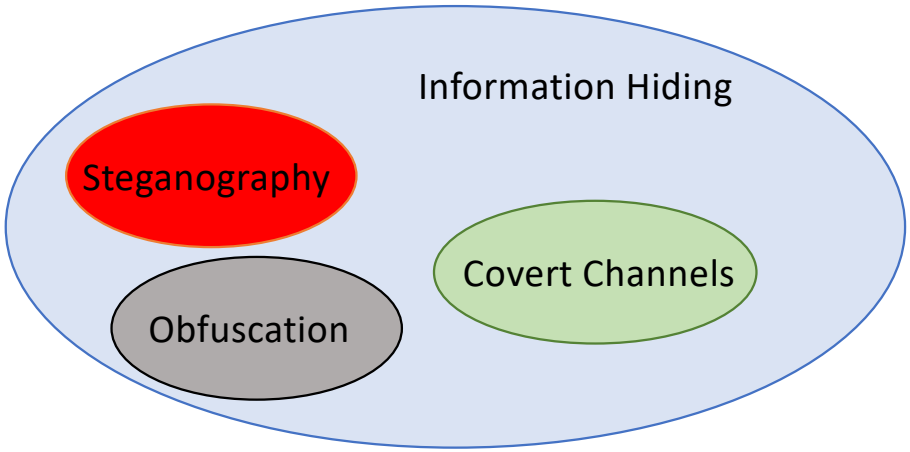


Steganographic Malware

…this is a “vague” definition

Motivations

- Some possible reasons (for the improved undetectability of malware) are:
  - Modular design for customization (e.g., Regin, Flamer, and Weevil)
  - Multistage loading (e.g., Regin, Stuxnet, and Duqu)
  - Cybercrime-as-a-Service models (e.g., Tox)
  - **Information Hiding** techniques (e.g., Platinum APT).



Stegomalware

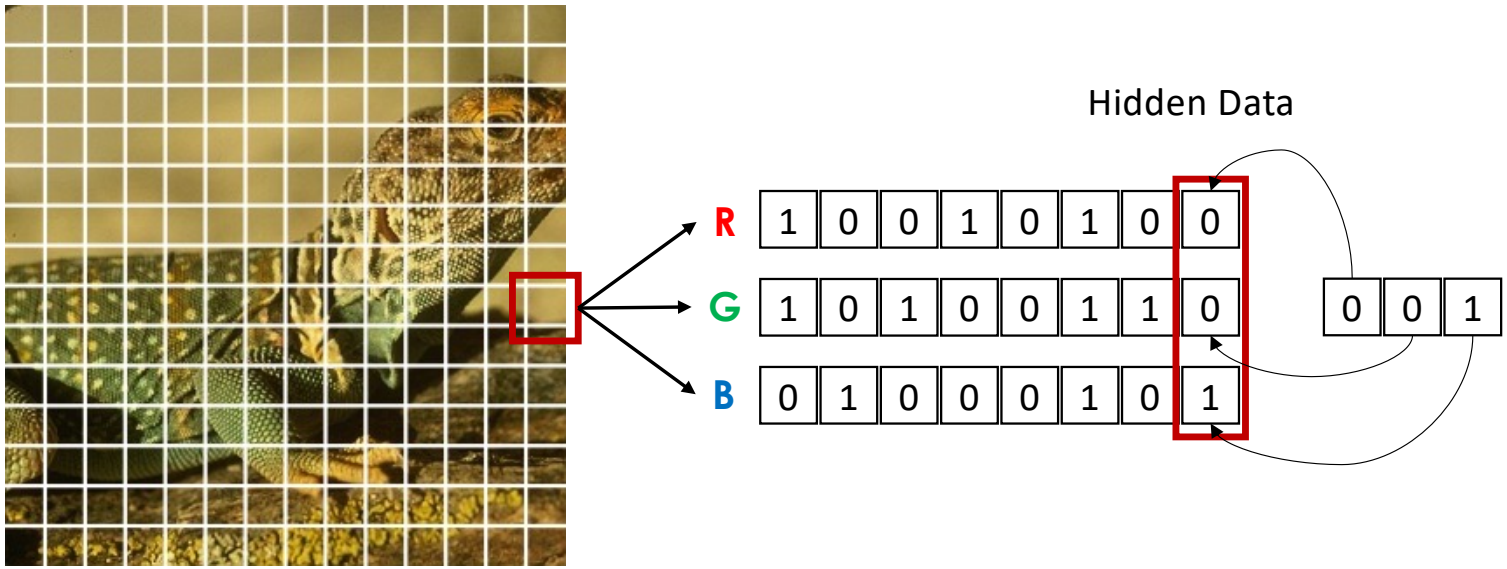
- Many researchers are starting to identify this class of threats as:
  - **Stegomalware**: steganographic malware
  - "borrowed" from works on mobile security and covert social botnets.
- A possible (common) definition:
  - Stegomalware is a malware using some form of steganography to remain undetected.
- Personally, I found it a bit reductive:
  - it narrows the scope too much
  - a bit ambiguous (Information Hiding vs steganography)
  - it is not only about detection (e.g., colluding applications).

G. Suarez-Tangil, J. Tapiador, P. Peris-Lopez, "Stegomalware: Playing Hide and Seek with Malicious Components in Smartphone Apps", 10th International Conference on Information Security and Cryptology, pp. 496–515, 2014
S. Nagaraja, A. Houmansadr, P. Piyawongwisal, V. Singh, P. Agarwal, B. Nikita, "Stegobot: A Covert Social Network Botnet", Information Hiding - 13th International Conference, pp. 299 – 313, 2011

LSB Steganography

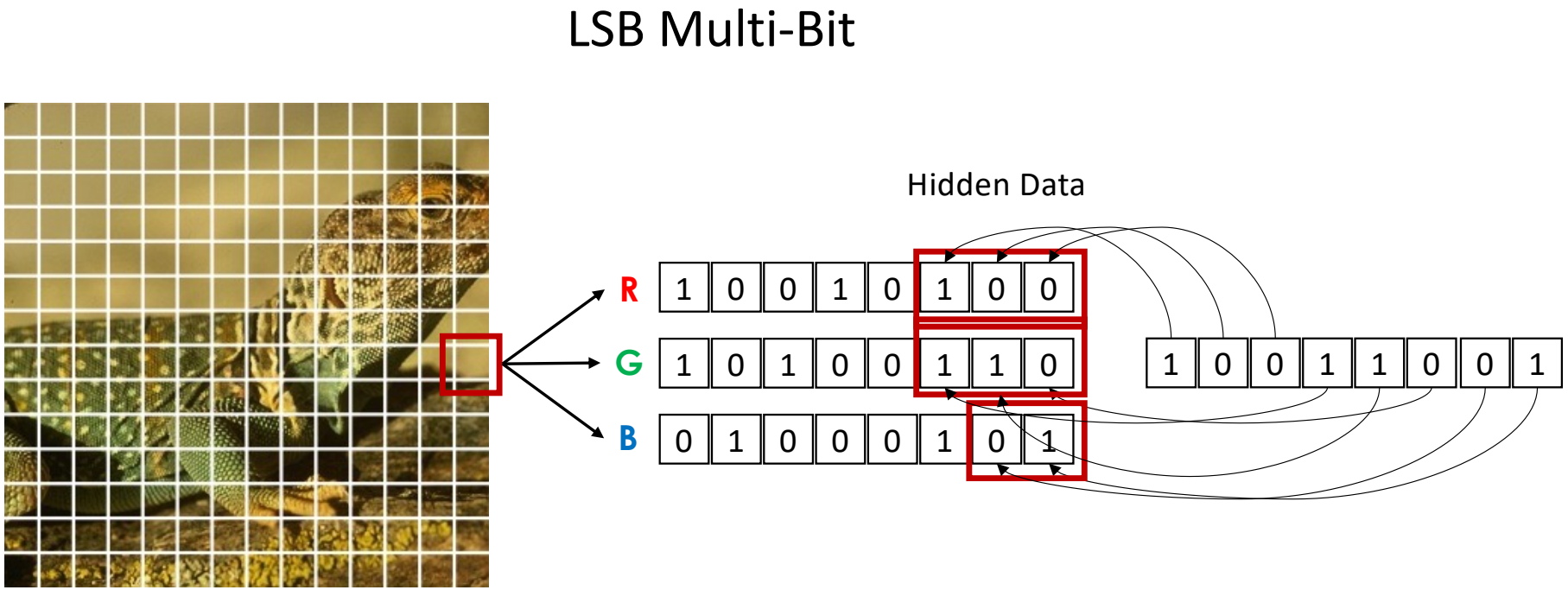
- Paradigmatic example: cloaking data in digital pictures

LSB “standard”



LSB Steganography

- Paradigmatic example: cloaking data in digital pictures



LSB Steganography - Example

- Data hidden by considering 1 Bit – Red Channel:

Original Image

Altered Image

Secret: 100 times “Summer School on Artificial Intelligence for a Secure Society”

LSB Steganography - Example

- Data hidden by considering 1 Bit – Red Channel:

Blue Channel
(Grayscale)

Original Image

Altered Image

LSB Steganography - Example

- Data hidden by considering 1 Bit – Red Channel:

Green Channel
(Grayscale)

Original Image

Altered Image

LSB Steganography - Example

- Data hidden by considering 1 Bit – Red Channel:

Red Channel
(Grayscale)

Original Image

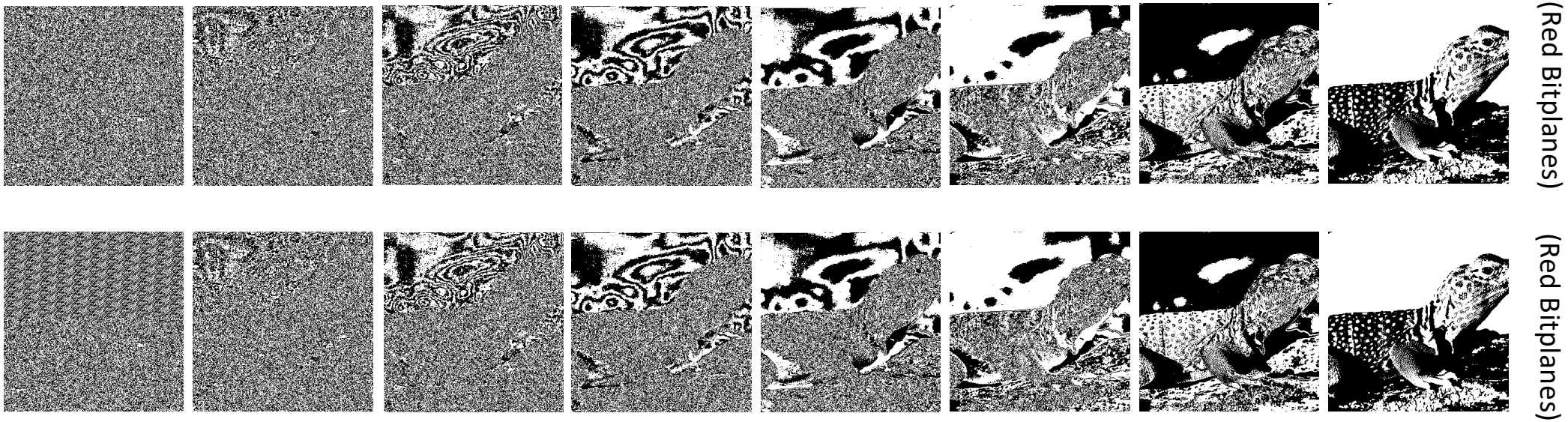
Altered Image

LSB Steganography - Example

- Data hidden by considering 1 Bit – Red Channel:

LSB

MSB

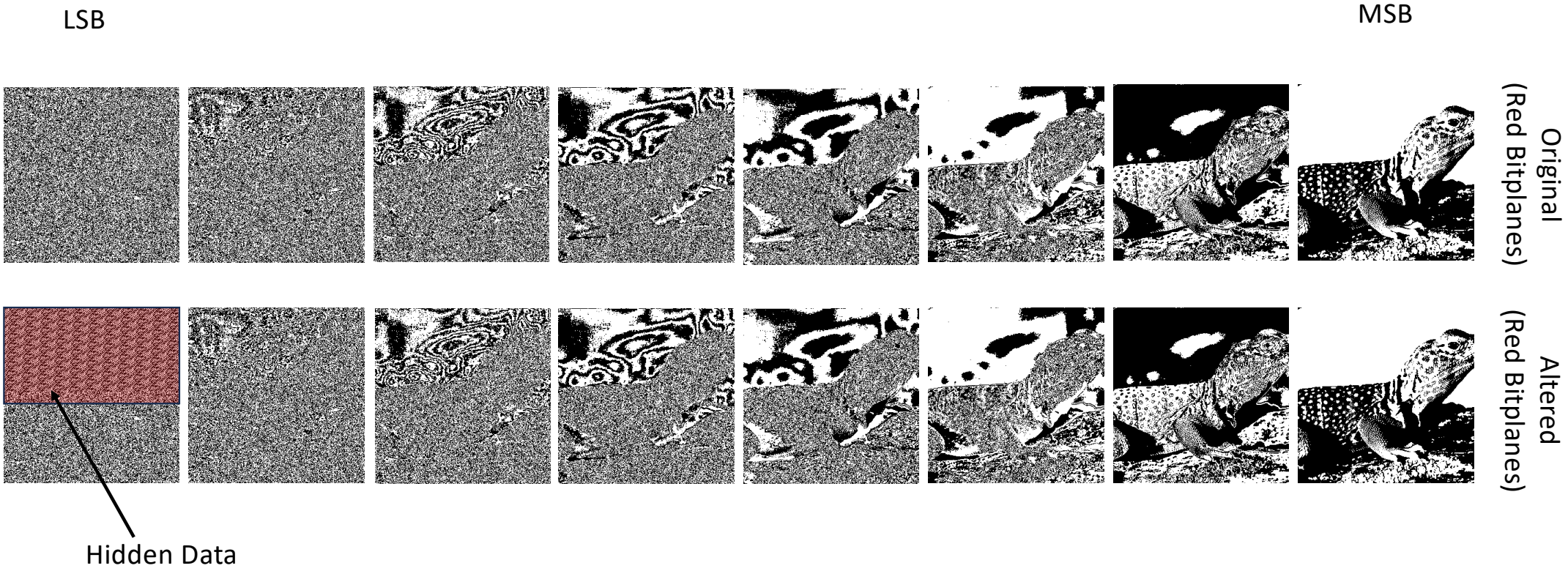


Original (Red Bitplanes)

Altered (Red Bitplanes)

LSB Steganography - Example

- Data hidden by considering 1 Bit – Red Channel:

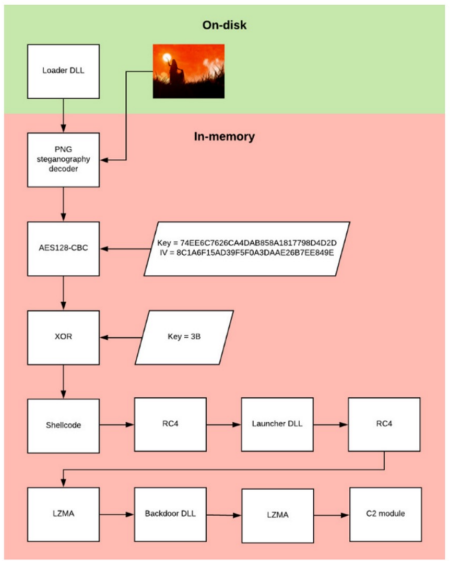


Malware Example - OceanLotus

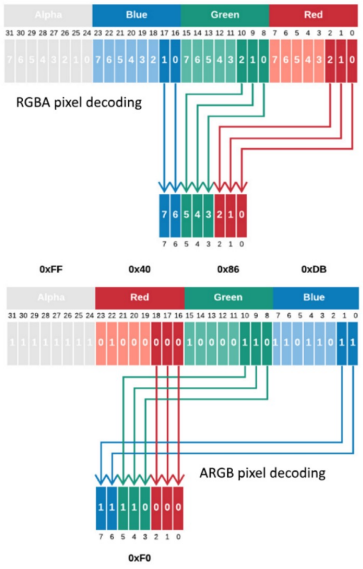
- Discovered in 2018/2019 (and used in several attack campaigns).
- The encrypted payload is concealed within a PNG image to elude detection (e.g., signature-based).
- Attack Phases:

Malware Example - OceanLotus

- Discovered in 2018/2019 (and used in several attack campaigns).
- The encrypted payload is concealed within a PNG image to elude detection (e.g., signature-based).
- Attack Phases:



Payload decoding process



Detection and Mitigation

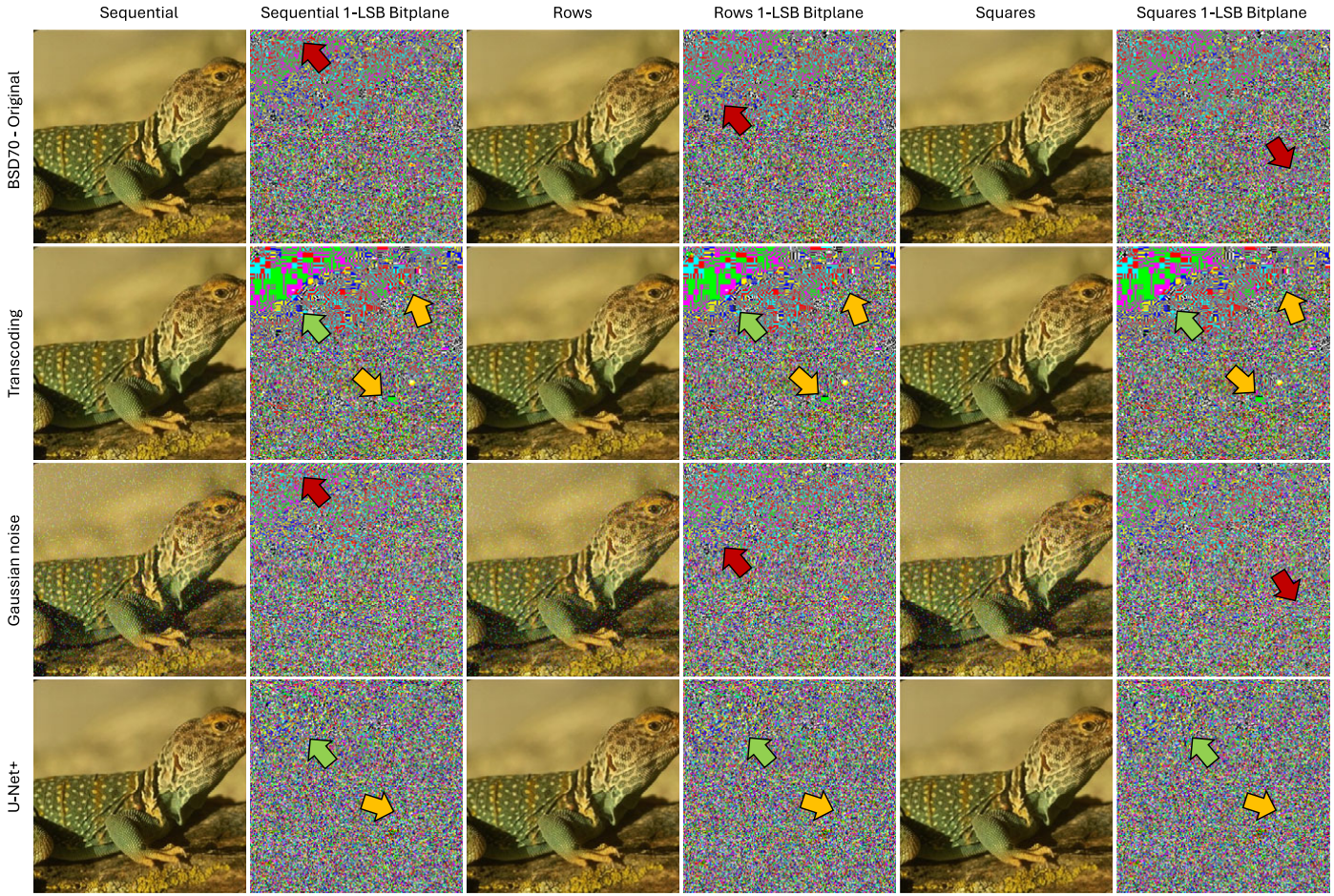
- Literature abounds in detection and mitigation techniques against multimedia steganography.
- Similar to the case of covert channels:
  - **statistical methods**: recognize traits that can reveal the presence of hidden data (e.g., color histogram)
  - **AI**: trained classifiers capable of distinguishing between legitimate and altered media.
- When detection is not possible, an approach is:
  - **sanitization**: try to remove the hidden payload or at least render it unusable for the attacker.

Detection and Mitigation

- Literature abounds in detection and mitigation techniques against multimedia steganography.
- Similar to the case of covert channels:
  - **statistical methods**: recognize traits that can reveal the presence of hidden data (e.g., color histogram)
  - **AI**: trained classifiers capable of distinguishing between legitimate and altered media.
- When detection is not possible, an approach is:
  - **sanitization**: try to remove the hidden payload or at least render it unusable for the attacker.

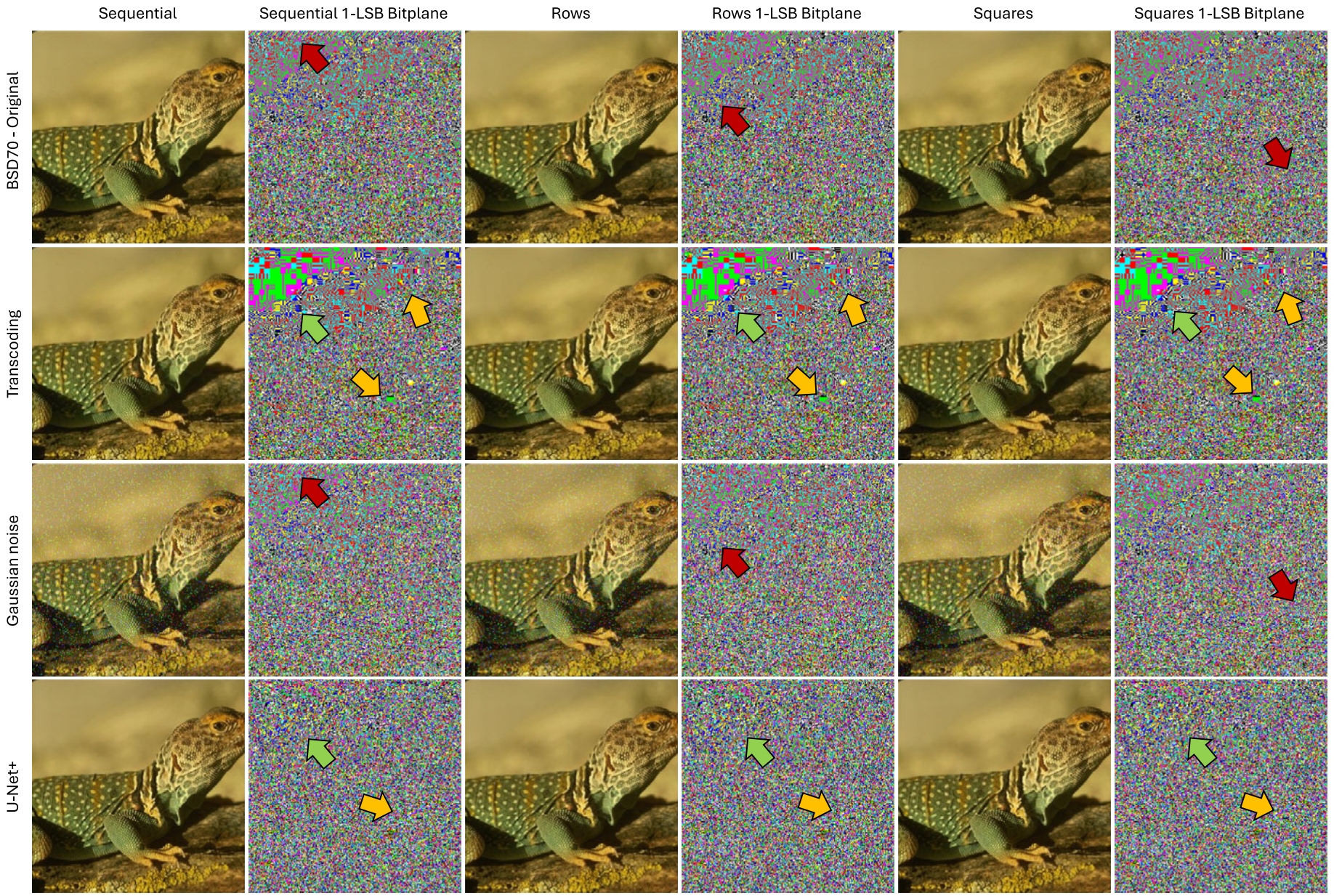
There are many pitfalls to consider.

Example - Image Sanitization



Legend:

- Malicious hidden content
- Artifacts of the sanitization
- Major differences


Sequential
Sequential 1-LSB Bitplane
Rows
Rows 1-LSB Bitplane
Squares
Squares 1-LSB Bitplane
BSD70 - Original
Transcoding
Gaussian noise
U-Net+

Side-Channel-Attacks

- A side channel is an attack that exploits “extra” information **leaked** by a system, rather than a weakness or a vulnerability.
- Usually, they require some form of **proximity** with the target.
- Great interest in 80/90 for attacking **cryptographic** mechanisms.
- Today widely used in:
  - networked/virtualized systems
  - software objects.

Side-Channel-Attacks

- A side channel is an attack that exploits “extra” information **leaked** by a system, rather than a weakness or a vulnerability.
- Usually, they require some form of **proximity** with the target.
- Great interest in 80/90 for attacking **cryptographic** mechanisms.
- Today widely used in:
  - networked/virtualized systems
  - software objects.

Examples

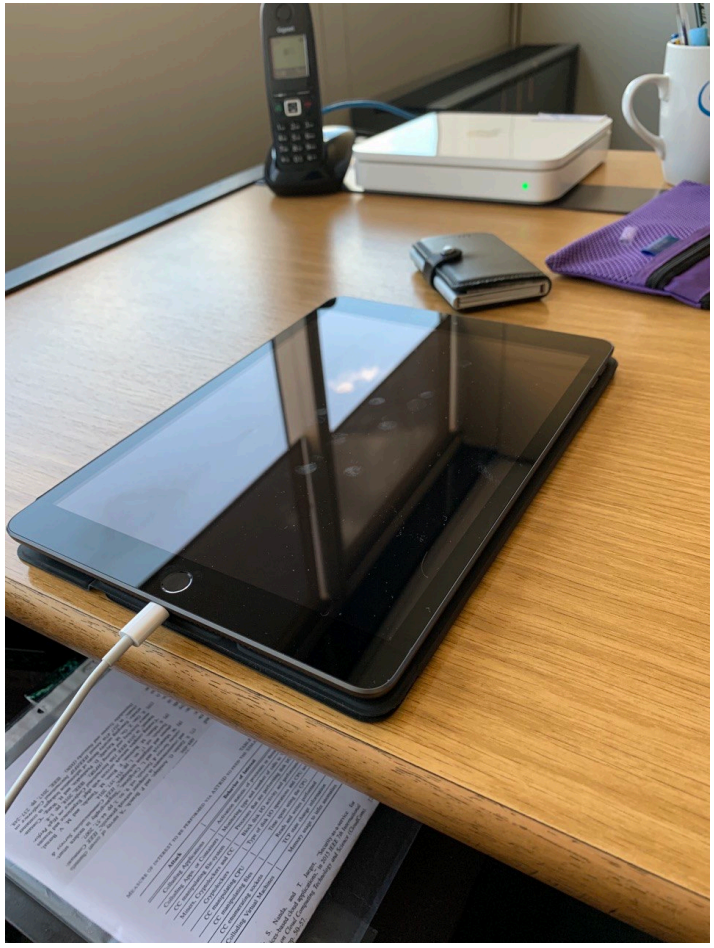
Temperature: to guess what a device or a program is doing.

Time: to understand whether traffic has been processed by a security tool or routed through a middlebox.

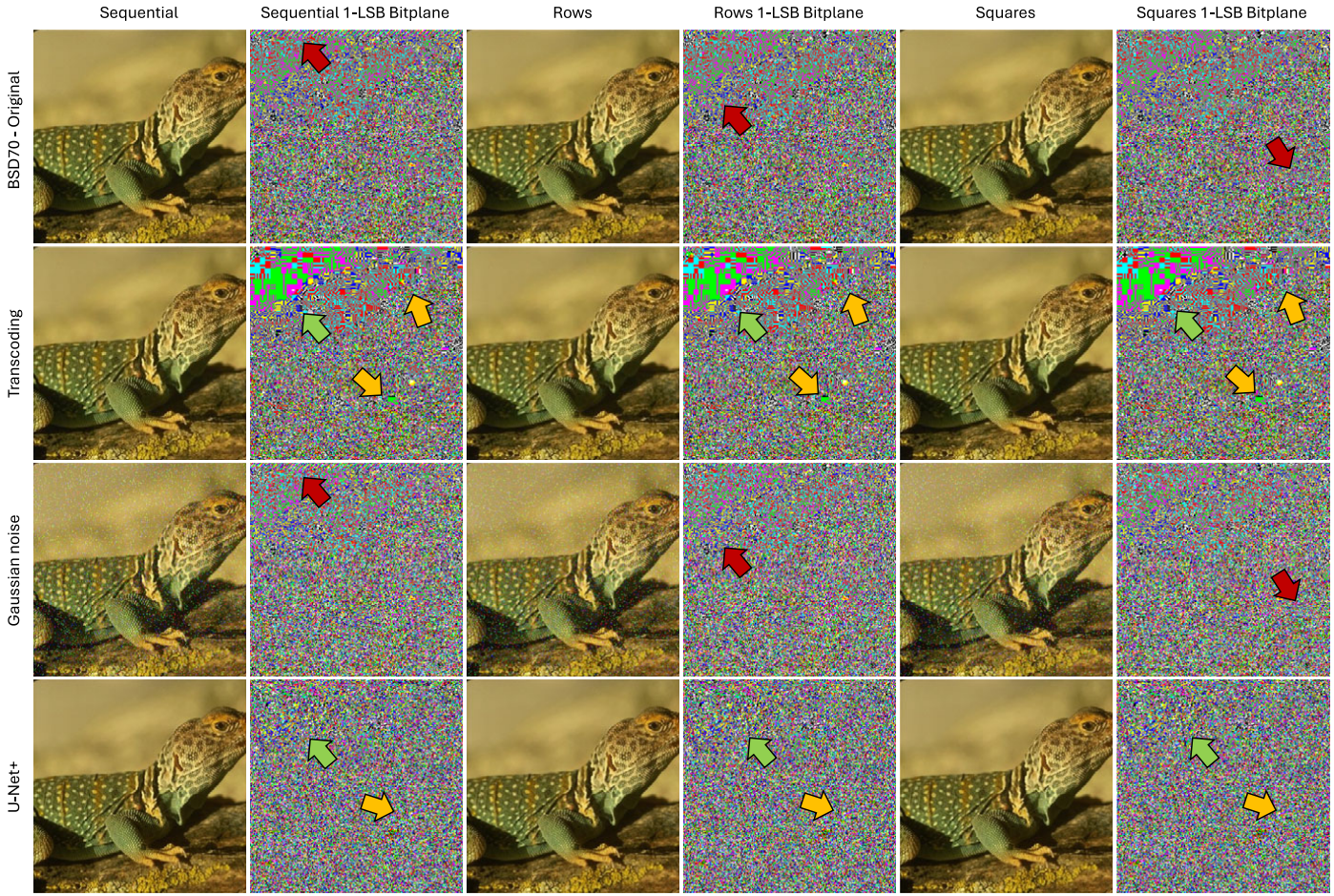
EM Emissions: to identify specific instruction patterns in GPUs.

Side-Channel-Attacks

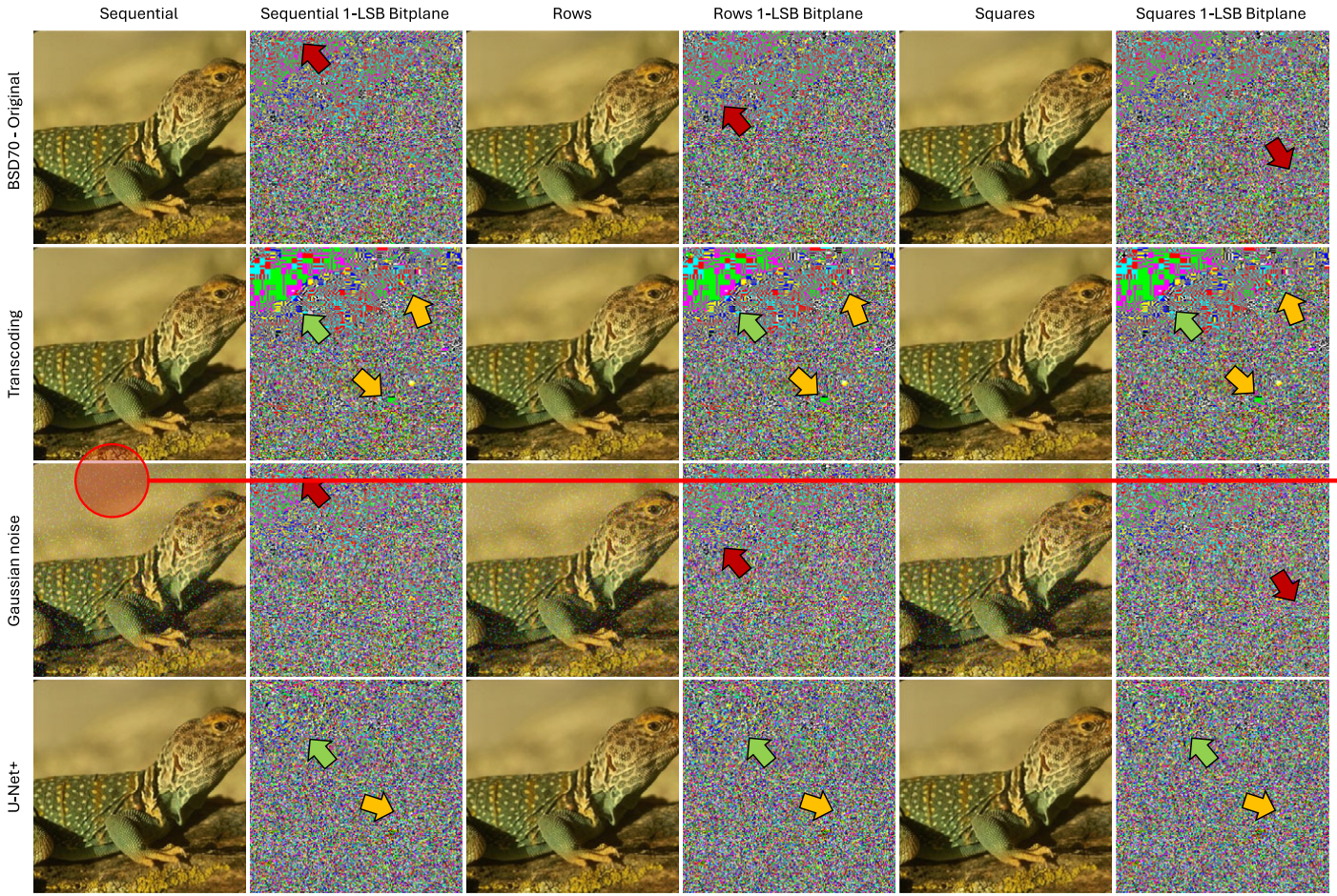
- A side channel is an attack that exploits “extra” information **leaked** by a system, rather than a weakness or a vulnerability.
- Usually, they require some form of **proximity** with the target.
- Great interest in 80/90 for attacking **cryptographic** mechanisms.
- Today widely used in:
  - networked/virtualized systems
  - software objects.



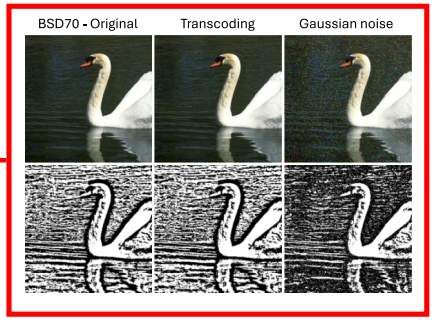
Example - Image Sanitization



Example - Image Sanitization

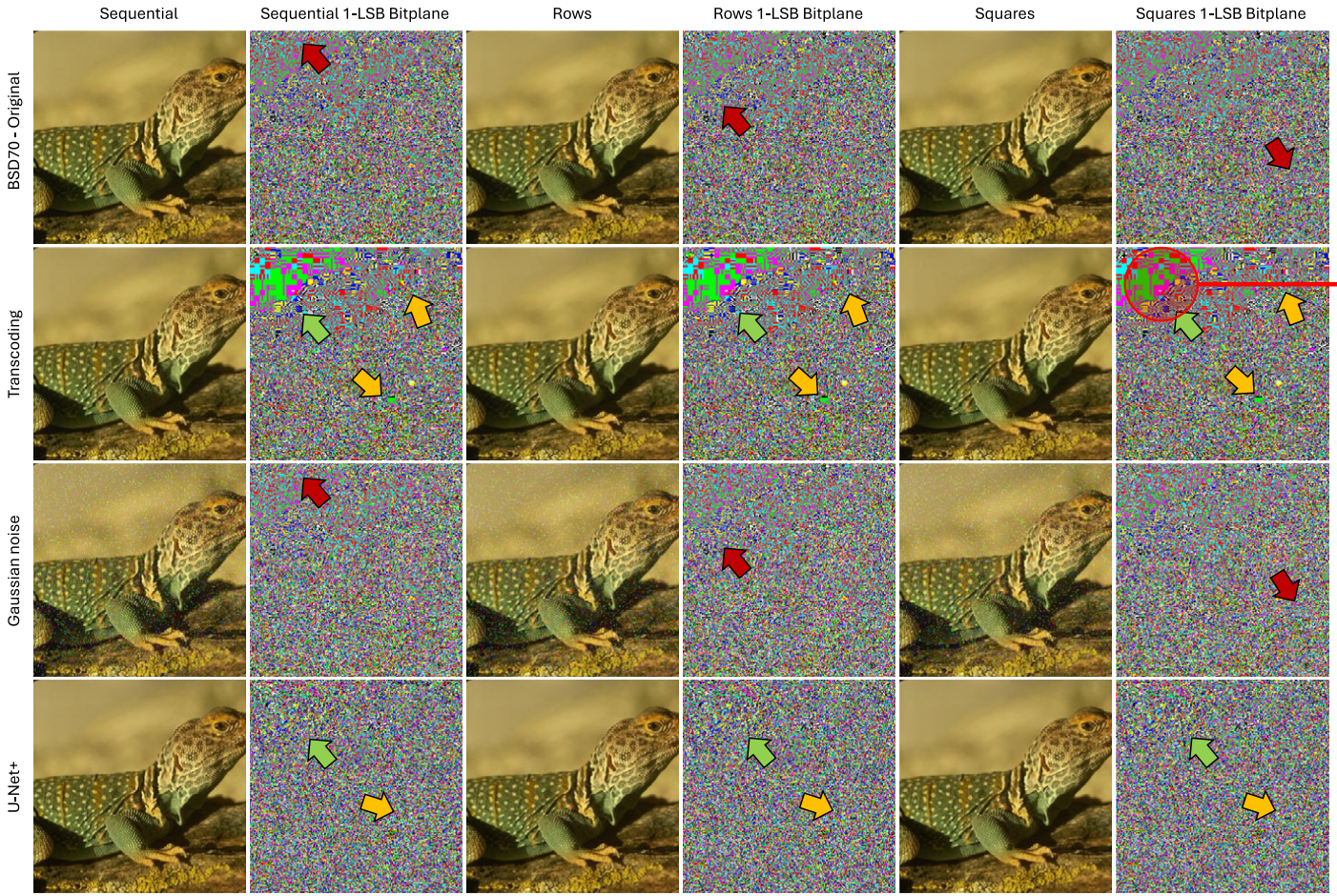


side-channel information



5 x 5 kernel filter

Example - Image Sanitization



side-channel information

Main Challenges

…against malware endowed with information hiding capabilities

Main Challenges

- There is **not** a **unique** hiding strategy and there is **not** a **general** defensive methodology.

Main Challenges

- There is **not** a **unique** hiding strategy and there is **not** a **general** defensive methodology.
- **Anticipating** ambiguities rather than fixing them later:
  - this requires formalization, by-design approaches, and SBoM-like tests
  - bring back hiding mechanisms to recurrent **patterns**
  - complex interplays difficult to capture “by hand”.

S. Wendzel, S. Zander, B. Fechner, C. Herdin, “Pattern-Based Survey and Categorization of Network Covert Channel Techniques”, ACM Computing Surveys, Vol. 47, No. 3, pp. 1-26, 2015

R. A. Kemmerer, “Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels”, ACM Transactions on Computer Systems, Vol. 1, No. 3, pp. 256-277, 1987

Main Challenges

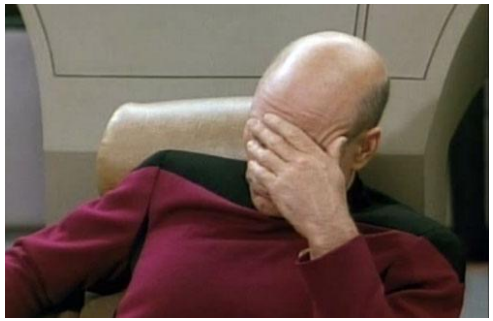
- There is **not** a **unique** hiding strategy and there is **not** a **general** defensive methodology.
- **Anticipating** ambiguities rather than fixing them later:
  - this requires formalization, by-design approaches, and SBoM-like tests
  - bring back hiding mechanisms to recurrent **patterns**
  - complex interplays difficult to capture “by hand”.
- Make existent approaches **scalable**:
  - especially for network traffic and multimedia contents
  - pay attention to **side-channel**-attacks
  - pursue generalization also in the **data gathering** phase.

S. Wendzel, S. Zander, B. Fechner, C. Herdin, “Pattern-Based Survey and Categorization of Network Covert Channel Techniques”, ACM Computing Surveys, Vol. 47, No. 3, pp. 1-26, 2015

R. A. Kemmerer, “Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channels”, ACM Transactions on Computer Systems, Vol. 1, No. 3, pp. 256-277, 1987

Conclusions

- The main challenges to address malware with information hiding capabilities:
  - carrier is **not** known *a priori* (e.g., network traffic and system-wide behaviors)
  - **heterogenous** set of protocols, software objects and data types
  - mixed **techniques** (storage or timing channels and steganography)
  - **GDPR-like** constraints
  - **scalability**
  - …
- Detection and mitigation are:
  - method-dependent
  - poorly generalizable.
- We should aim at **preventing** and **eliminating** ambiguities.



Selected References

A curated list of malware using information hiding and covert channels: https://github.com/lucacav/steg-in-the-wild

W. Mazurczyk, **L. Caviglione**, “Steganography in Modern Smartphones and Mitigation Techniques”, IEEE Communications Surveys & Tutorials, Vol. 17, No. 1, pp. 334-357, First Quarter 2015.

W. Mazurczyk, **L. Caviglione**, “Information Hiding as a Challenge for Malware Detection”, IEEE Security & Privacy, Vol. 13, No. 2, pp. 89-93, March-April 2015.

W. Mazurczyk, **L. Caviglione**, “Cyber Reconnaissance Techniques: Evolution and Countermeasures”, Communications of the ACM, Vol. 64, N. 3, pp. 86 - 95, March 2021.

L. Caviglione, W. Mazurczyk, “Never Mind the Malware, Here’s the Stegomalware”, IEEE Security & Privacy, Vol. 20, No. 5, pp. 101-106, September-October 2022.

M. Zuppelli, M. Repetto, **L. Caviglione**, E. Cambiaso, “Information Leakages of Docker Containers: Characterization and Mitigation Strategies”, IEEE 9th International Conference on Network Softwarization, Madrid, Spain, pp. 462-467, June 2023.

P. Zorawski, **L. Caviglione**, W. Mazurczyk, “A Long-term Perspective of the Internet Susceptibility to Covert Channels”, IEEE Communications Magazine, Vol. 61, No. 10, pp. 171 - 177, October 2023.

E. Cambiaso, **L. Caviglione**, M. Zuppelli, “DockerChannel: A Framework for Evaluating Information Leakages of Docker Containers”, SoftwareX, Vol. 24, pp. 1-7, December 2023.

A. Liguori, M. Zuppelli, D. Gallo, M. Guarascio, **L. Caviglione**, “A Deep Learning-based Approach for Stegomalware Sanitization in Digital Images”, Journal of Intelligent Information Systems, pp. 1 - 24, March 2025.

Thank You!
Questions?

https://github.com/lucacav