Consiglio Nazionale delle Ricerche

Covert Channels in a Nutshell

Attack Models and Main Challenges

Luca Caviglione

Institute for Applied Mathematics and Information Technologies

luca.caviglione@cnr.it

University of Padova – Advanced Topics in Computer and Network Security
Padova, December 13, 2023

Outline

- Motivations
- (a quick introduction to) Covert Channels
- Network Covert Channels
- Local Covert Channels
- Emerging Usages of Covert Channels
- Research Challenges
- Conclusions
- Collaborations and Research Projects
- 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

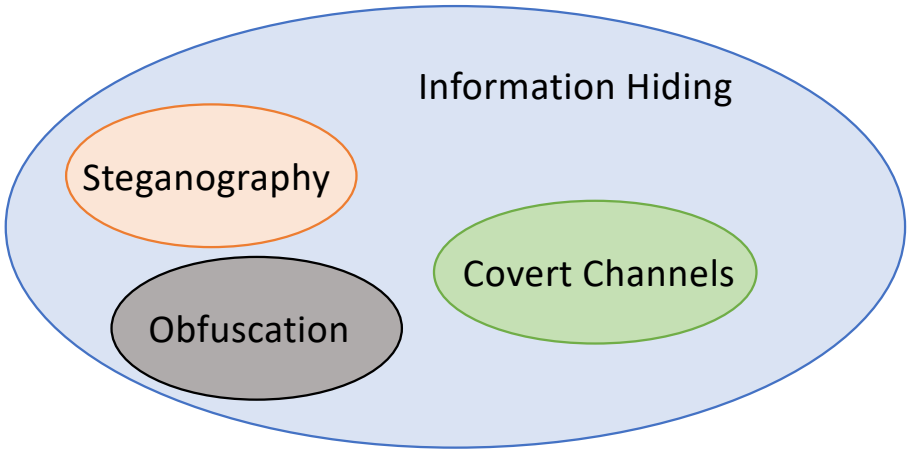
Motivations

- 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!

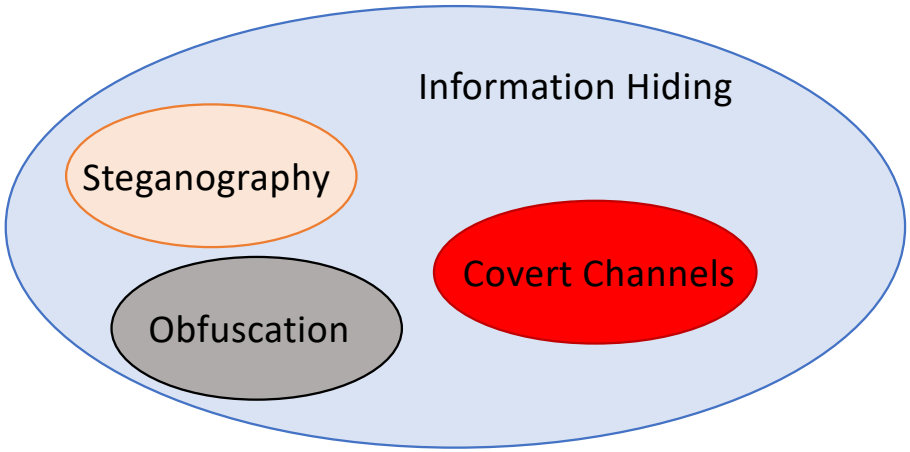
Motivations

- 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).

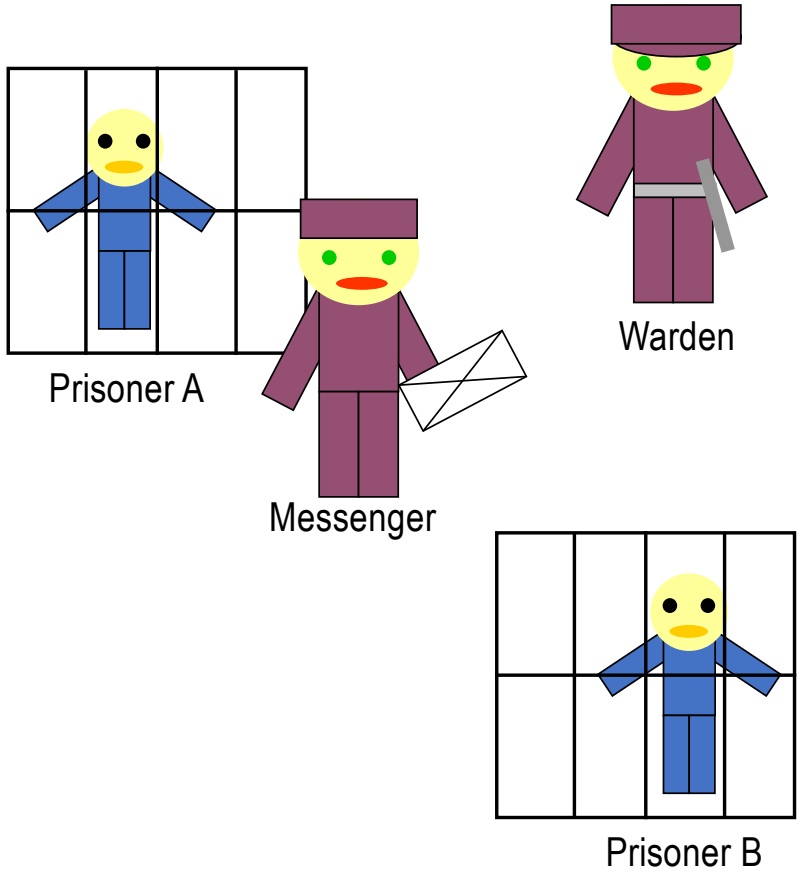


Covert Channels - In Nature

- **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

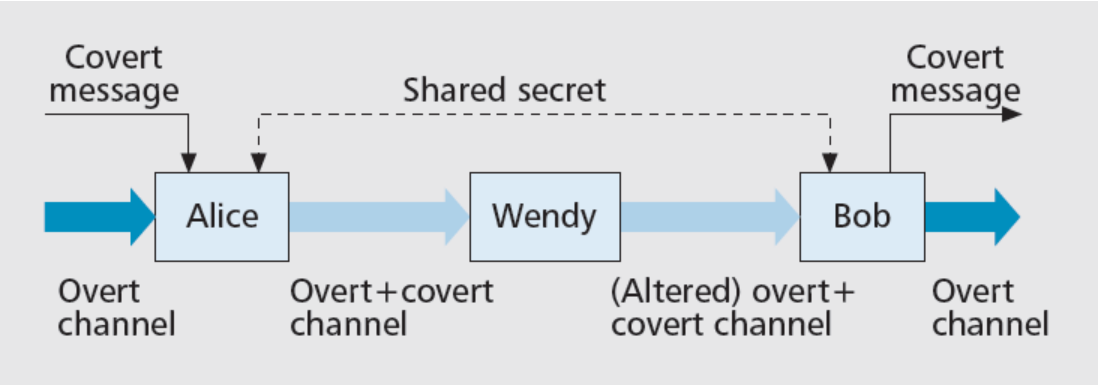
- Modeled by **Simmons** (1984) as the “Prisoners’ Problem and the Subliminal Channel”.



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

Covert Channels - Models

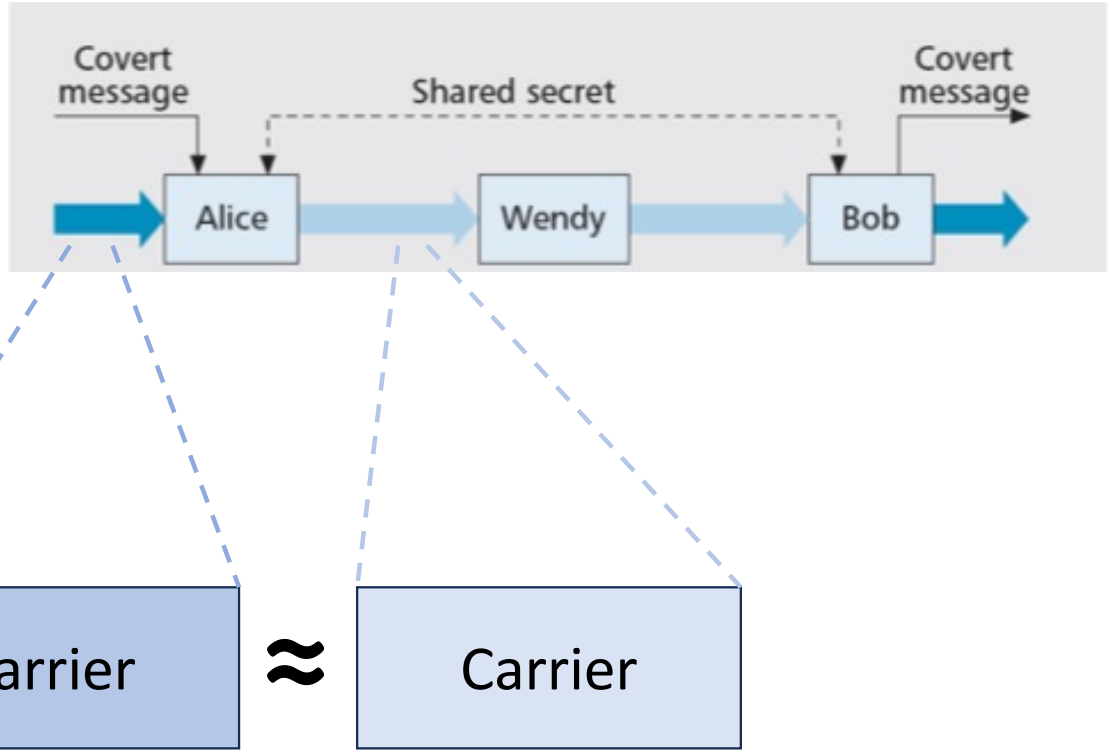
- Modeled by **Simmons** (1984) as the "Prisoners' Problem and the Subliminal Channel".
- According to **Lampson** (1973):
  - *"[channels] not intended for information transfer at all, such as the service program's effect on the system load*".



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

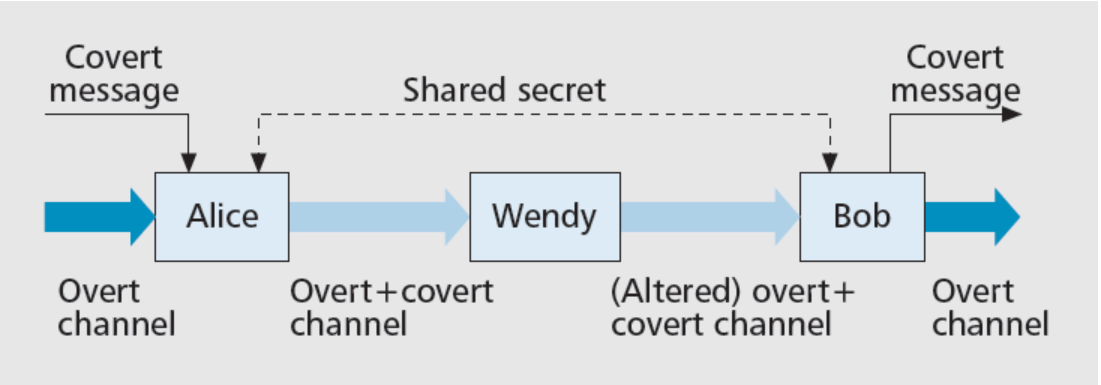
- Modeled by **Simmons** (1984) as the "Prisoners' Problem and the Subliminal Channel".
- According to **Lampson** (1973):
  - *"[channels] not intended for information transfer at all, such as the service program's effect on the system load*".



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

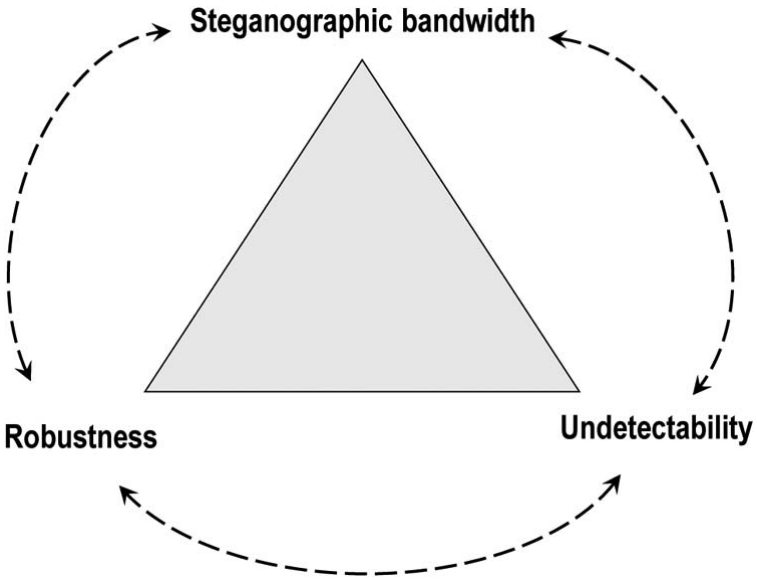
Information Hiding: information is difficult to **notice**
Cryptography: information is difficult to **comprehend**

Covert Channels - Metrics

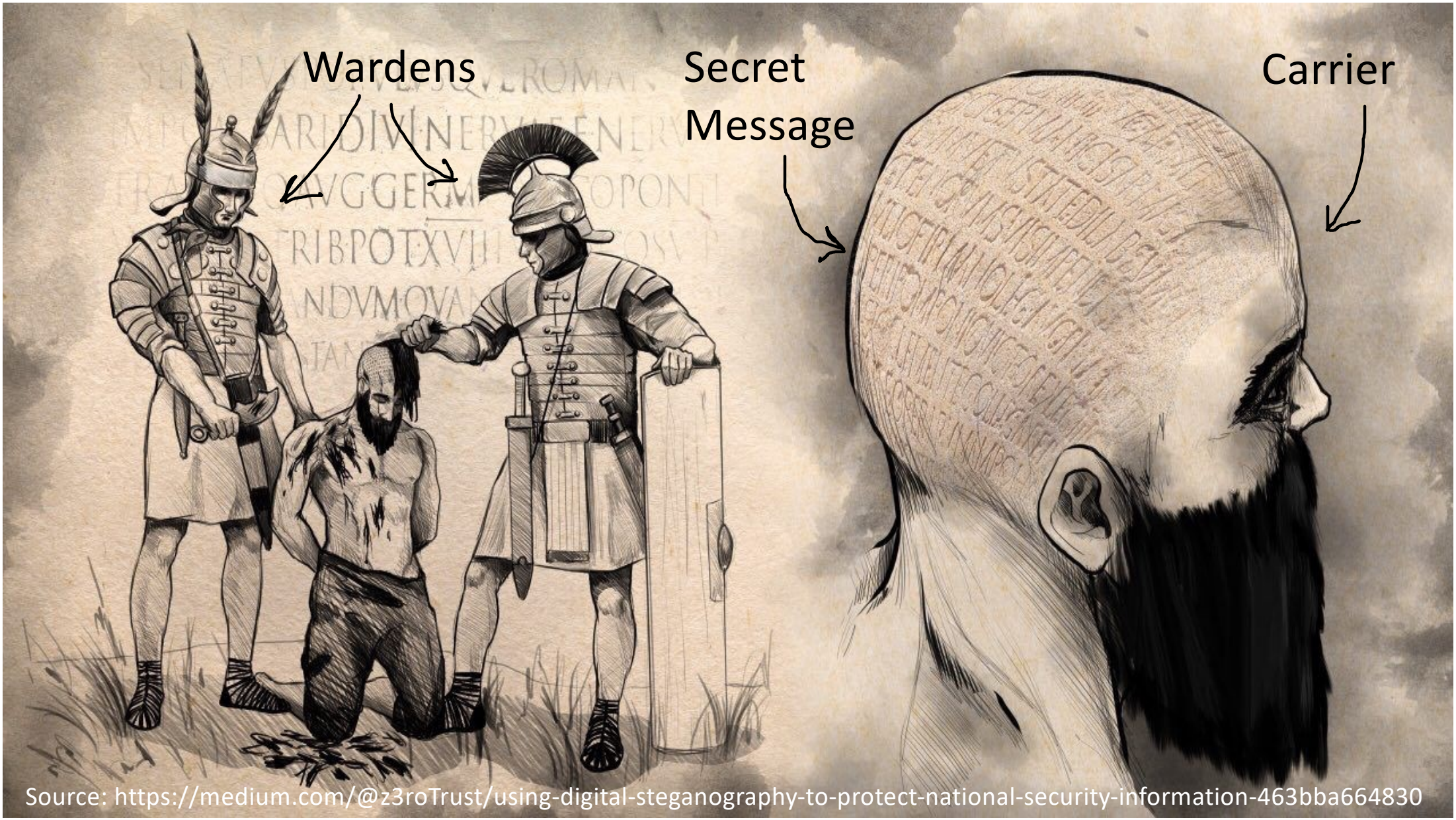
- Covert communications are usually characterized via three metrics:
  - **bandwidth**: the amount of secret data that can be sent per time unit when using a particular method
  - **undetectability**: the inability to detect secret data within a certain carrier
  - **robustness:** the amount of alteration 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

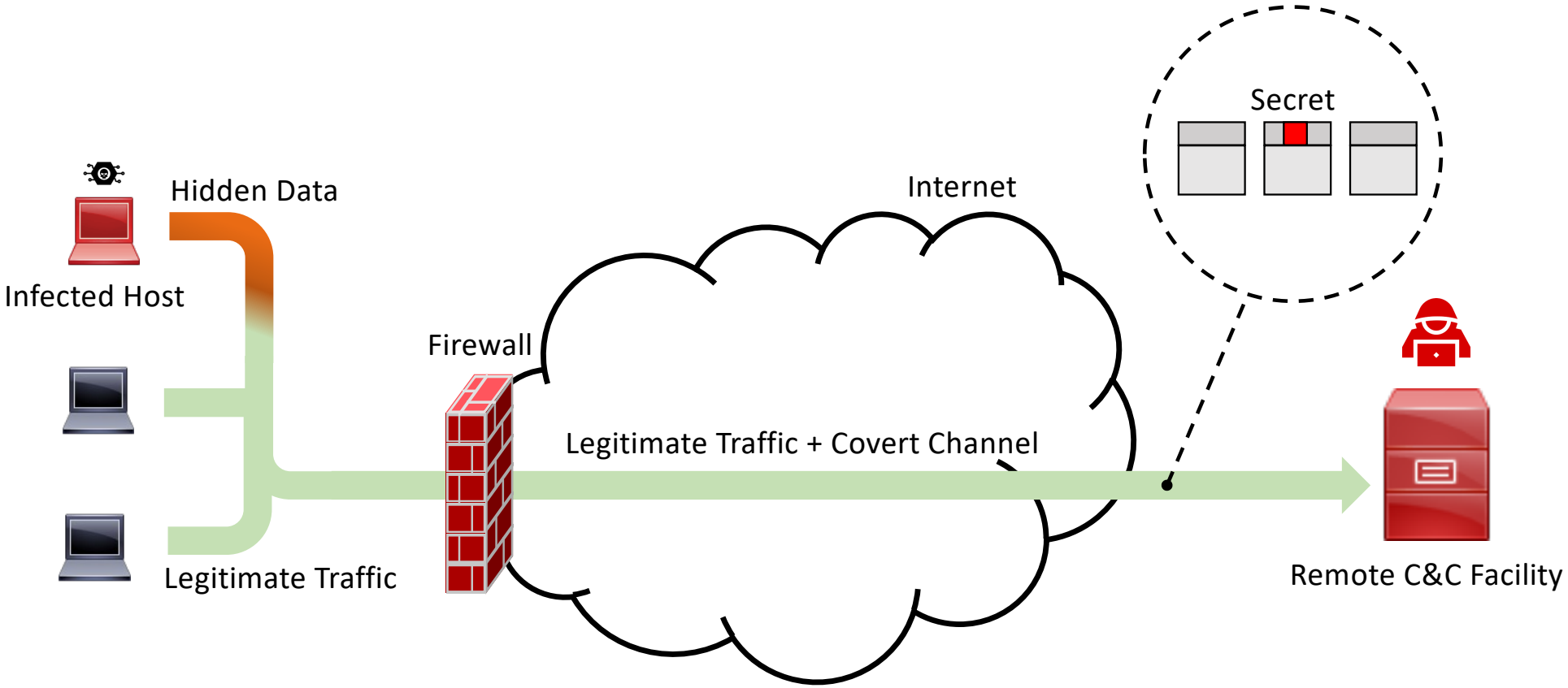


Source: https://medium.com/@z3roTrust/using-digital-steganography-to-protect-national-security-information-463bba664830

Network Covert Channels

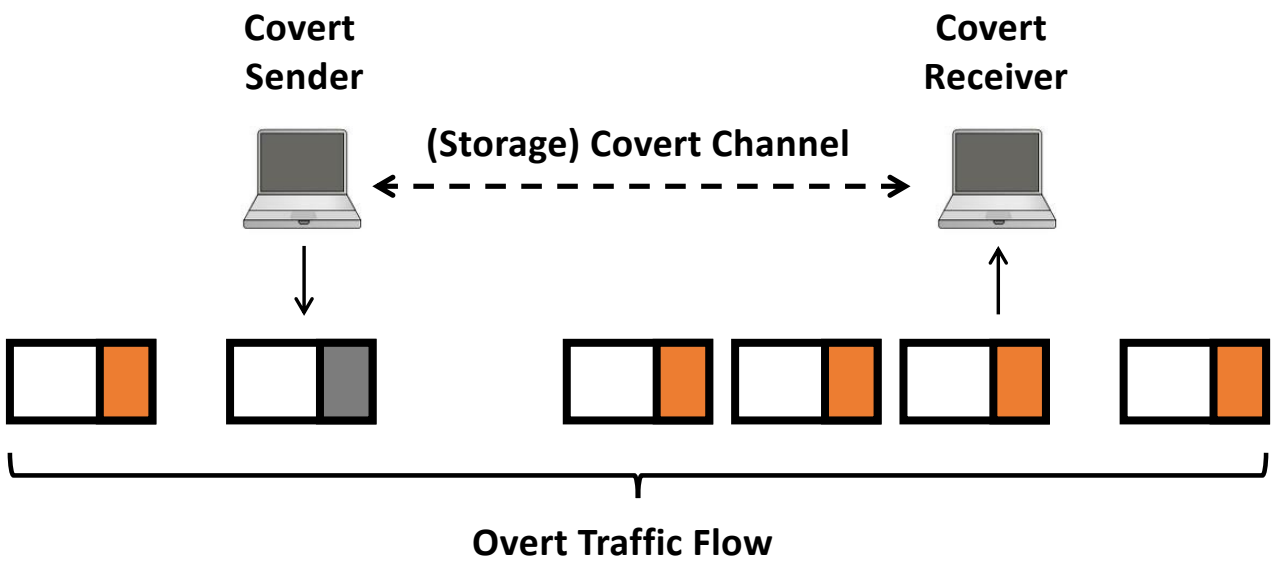
…also denoted as network steganography or out-of-band channels

Covert Channels in Network Traffic

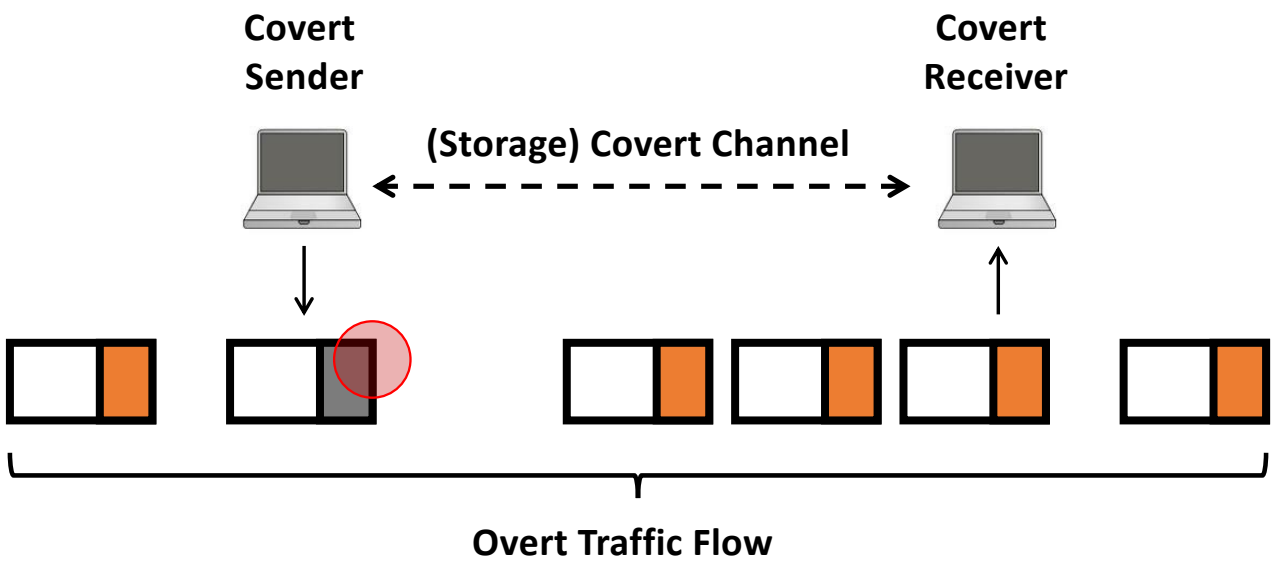


Attack Model

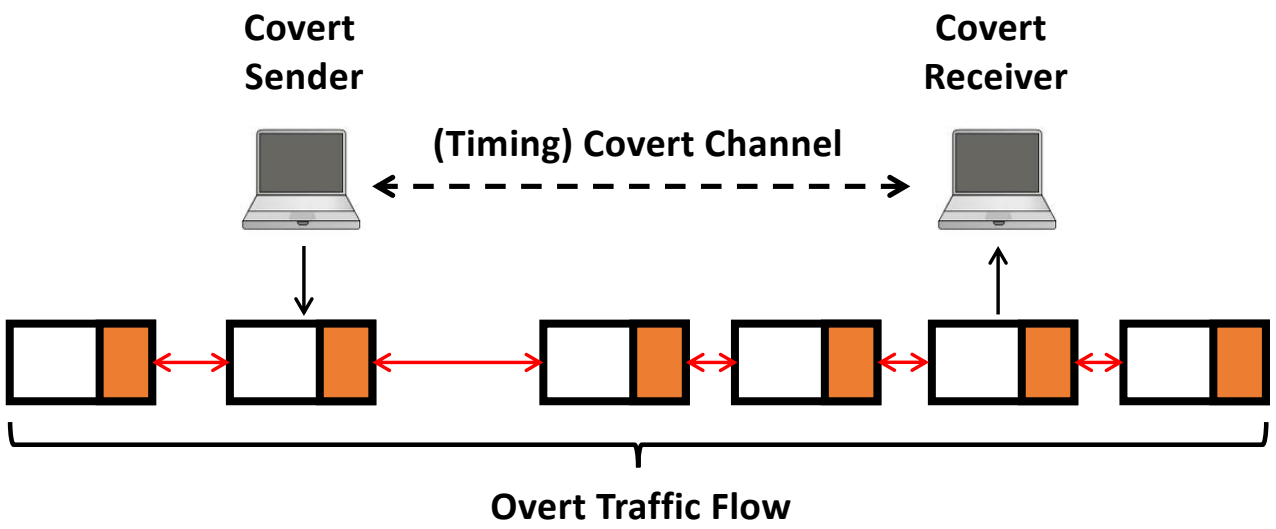
Covert Channels in Network Traffic



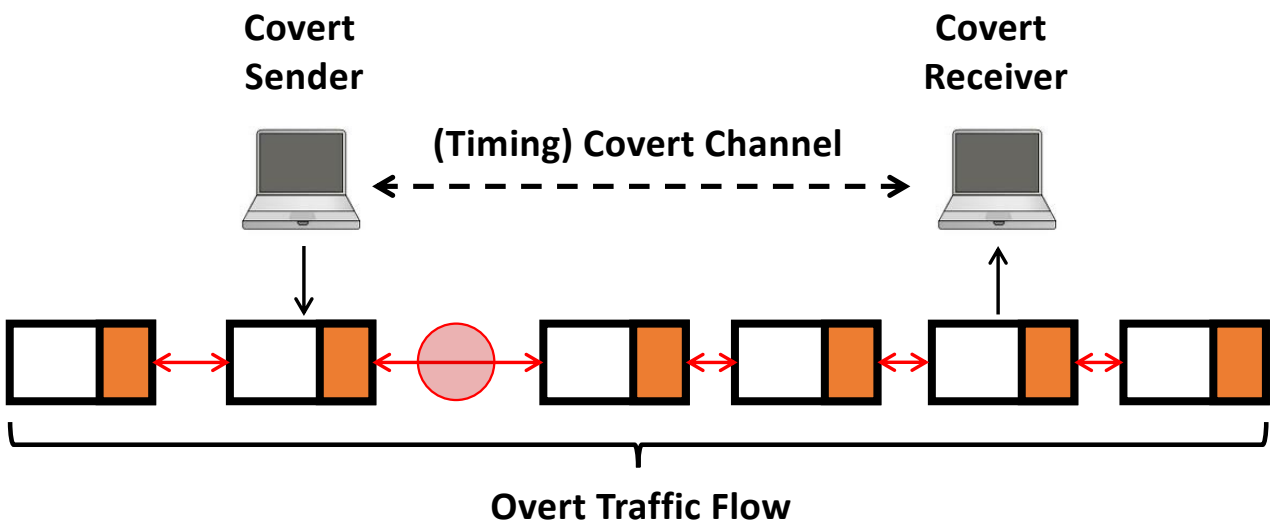
Covert Channels in Network Traffic



Covert Channels in Network Traffic

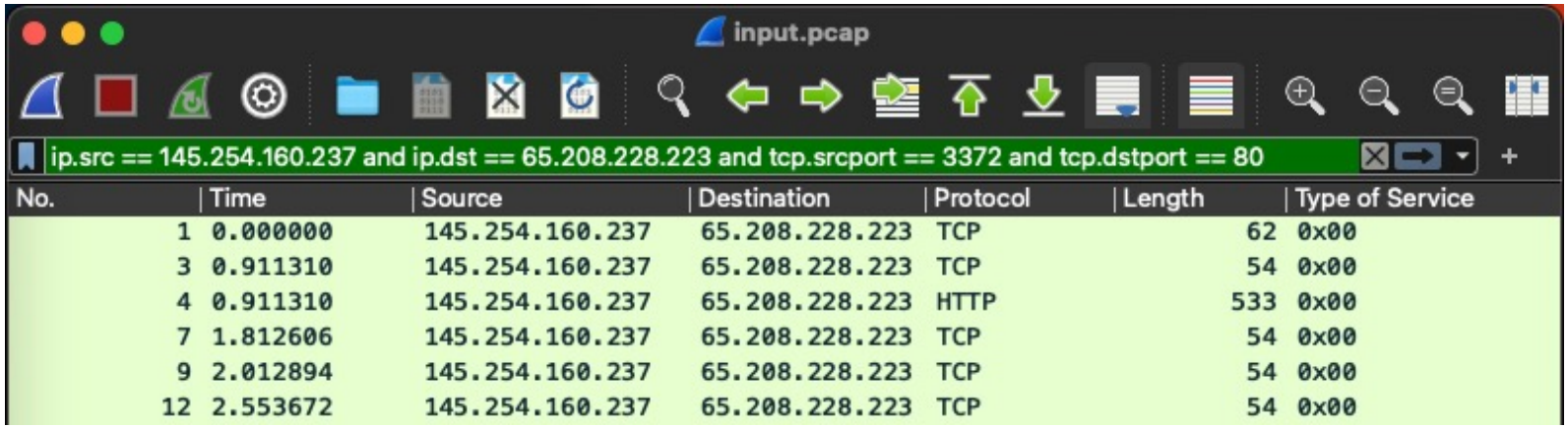


Covert Channels in Network Traffic



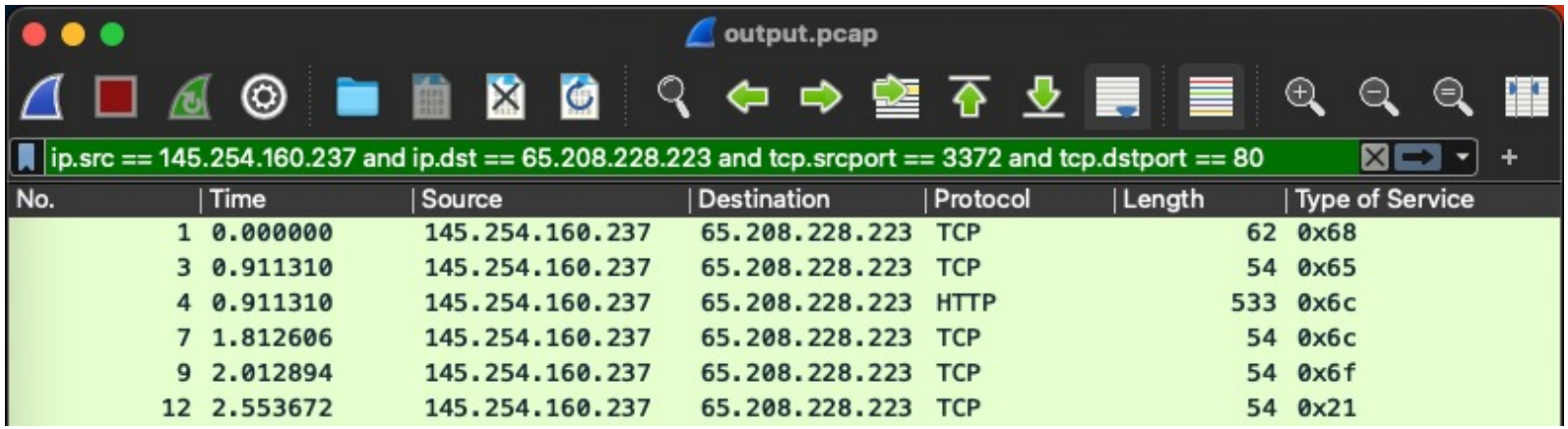
Example - Storage Channel in Type of Service

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

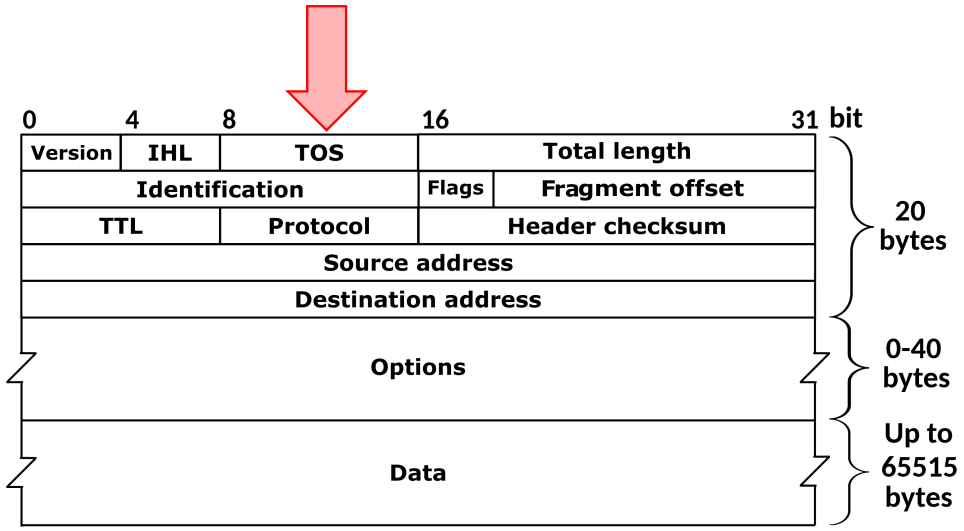
Overt + Covert



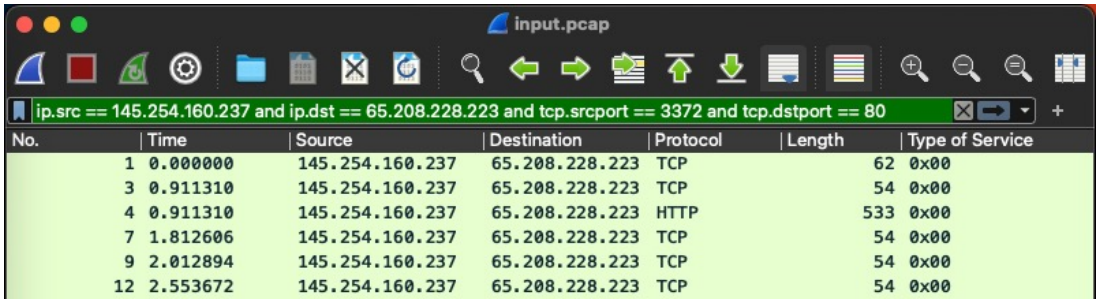
No.	Time	Source	Destination	Protocol	Length	Type of Service
1	0.000000	145.254.160.237	65.208.228.223	TCP	62	0x68
3	0.911310	145.254.160.237	65.208.228.223	TCP	54	0x65
4	0.911310	145.254.160.237	65.208.228.223	HTTP	533	0x6c
7	1.812606	145.254.160.237	65.208.228.223	TCP	54	0x6c
9	2.012894	145.254.160.237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

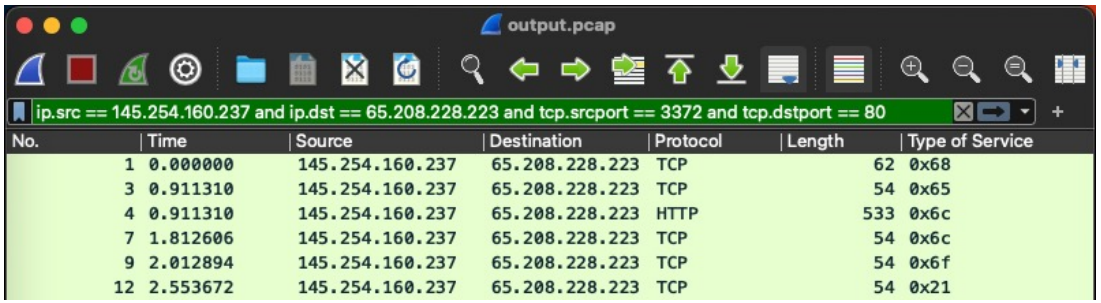
Example - Storage Channel in Type of Service



Overt



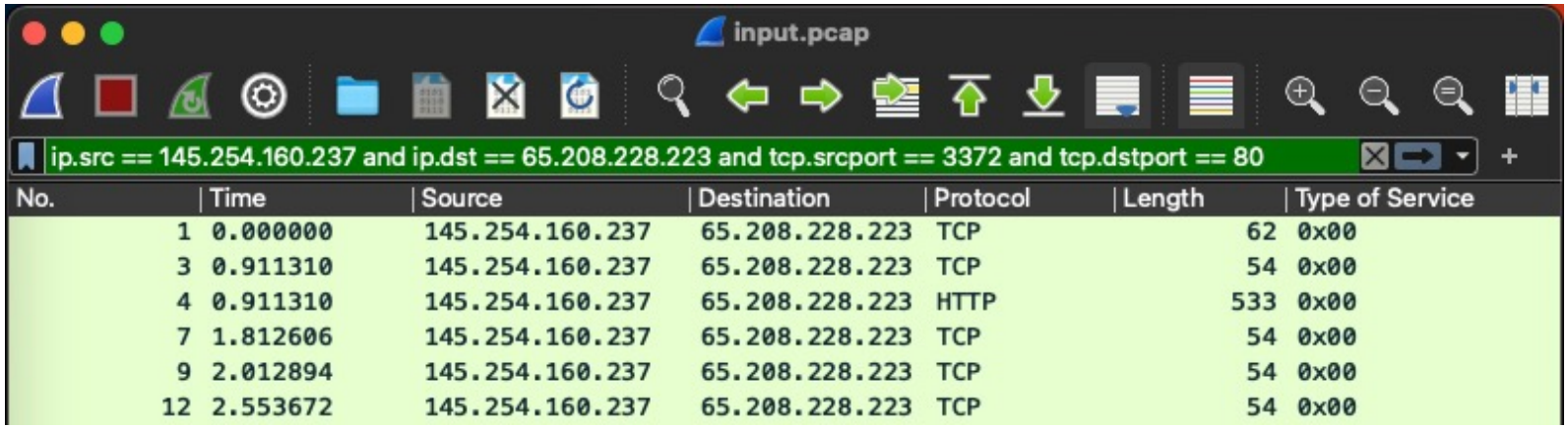
Overt + Covert



https://github.com/Ocram95/pcap_injector

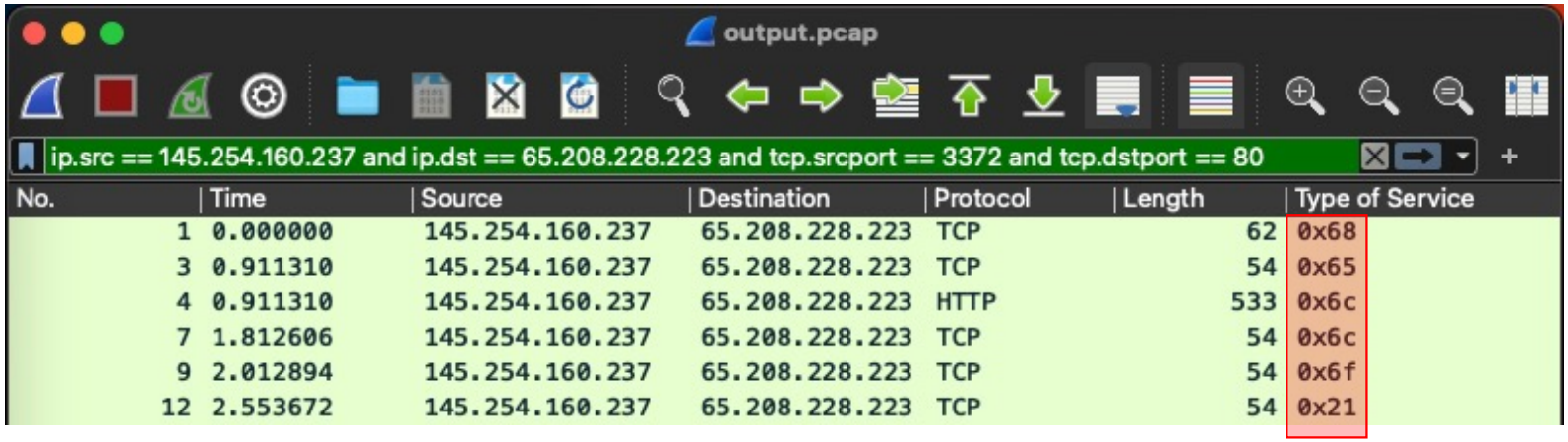
Example - Storage Channel in Type of Service

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



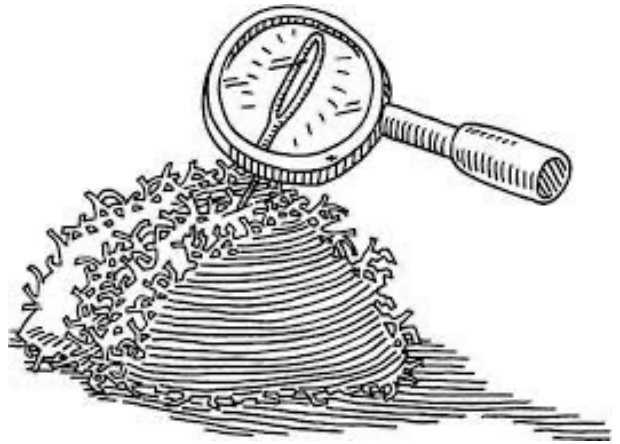
No.	Time	Source	Destination	Protocol	Length	Type of Service
1	0.000000	145.254.160.237	65.208.228.223	TCP	62	0x68
3	0.911310	145.254.160.237	65.208.228.223	TCP	54	0x65
4	0.911310	145.254.160.237	65.208.228.223	HTTP	533	0x6c
7	1.812606	145.254.160.237	65.208.228.223	TCP	54	0x6c
9	2.012894	145.254.160.237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

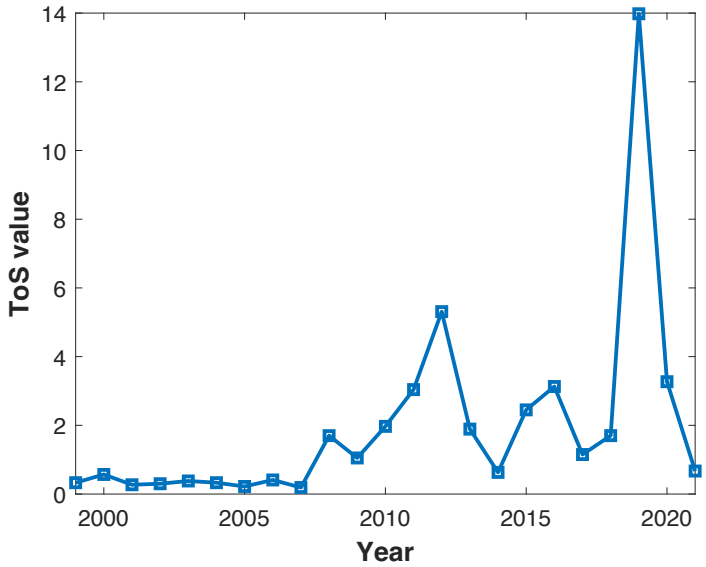
Example - Storage Channel in Type of Service

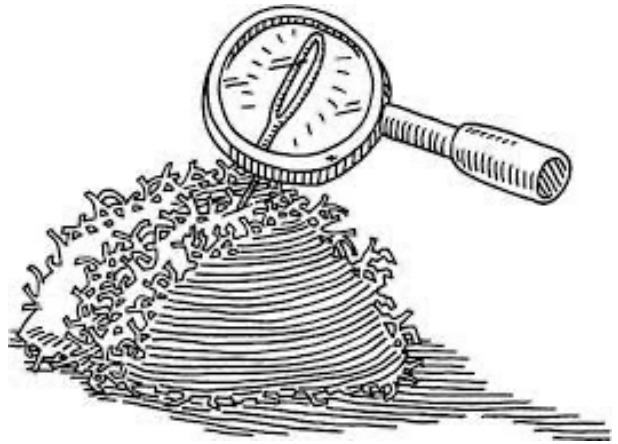
- To be effective, a (network) covert channel should:
  - use a **popular** carrier
  - not reveal the presence of the channel via **anomalous** behaviors
  - **not disrupt** the functionality of a protocol or a service.



Example - Storage Channel in Type of Service

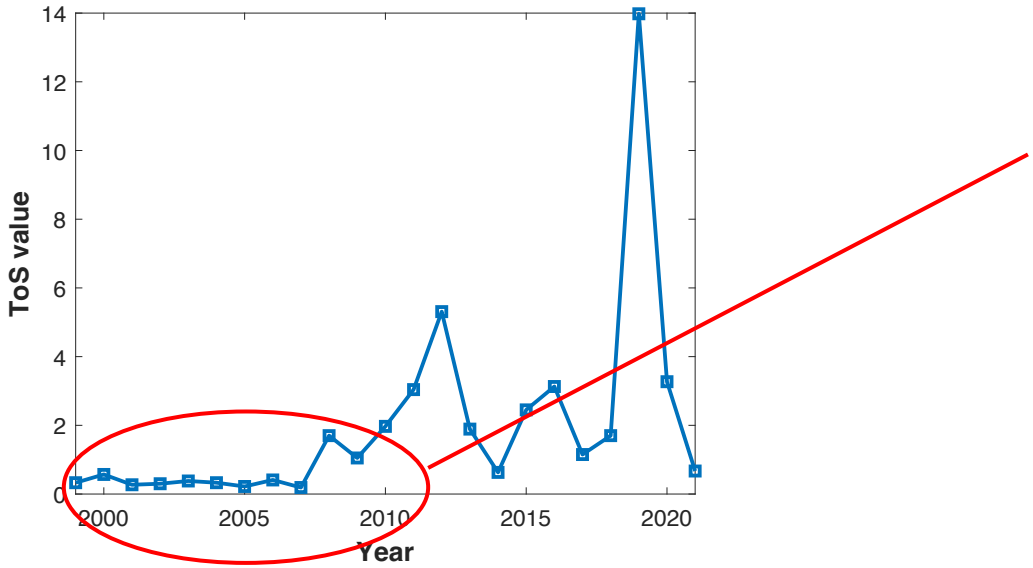
- To be effective, a (network) covert channel should:
  - use a **popular** carrier
  - not reveal the presence of the channel via **anomalous** behaviors
  - **<u>not disrupt</u>** the functionality of a protocol or a service.





Example - Storage Channel in Type of Service

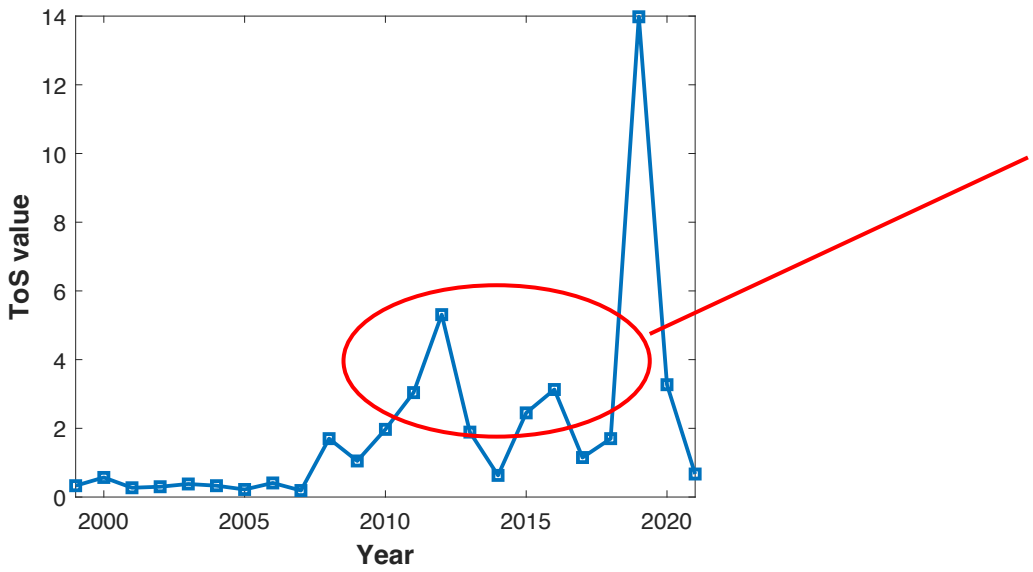
- To be effective, a (network) covert channel should:
  - use a **popular** carrier
  - not reveal the presence of the channel via **anomalous** behaviors
  - **<u>not disrupt</u>** the functionality of a protocol or a service.



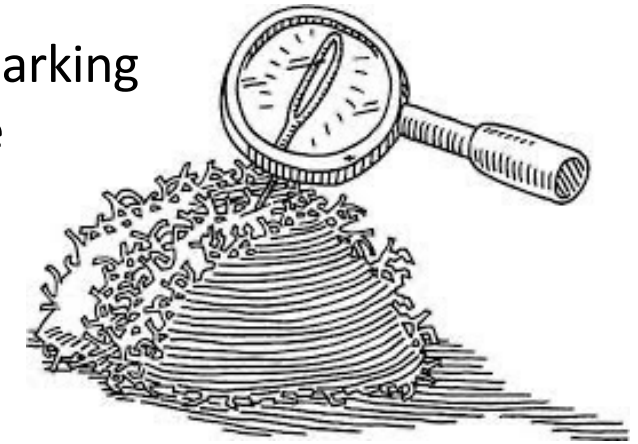
Only few values, thus making the presence of arbitrary data easily detectable.

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.



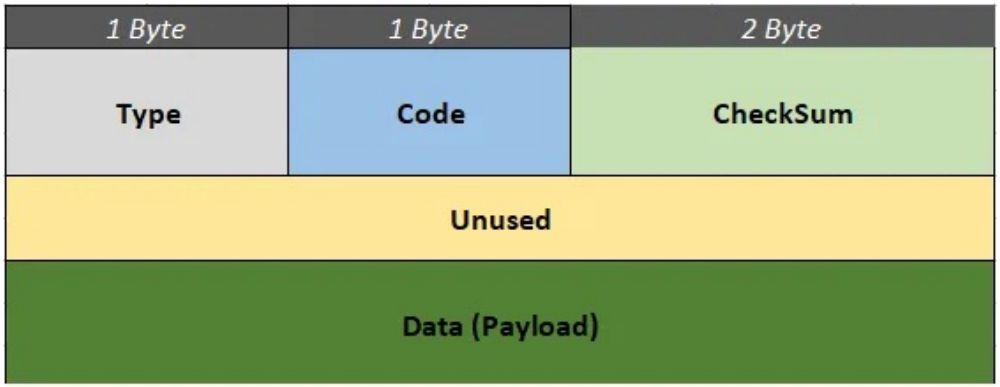
Malware Examples - PlugX 2.0

- Discovered in 2014 (update of the PlugX RAT).
- It uses a custom algorithm for encryption and hides C&C communications in ICMP, HTTP, and DNS 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, HTTP, and DNS 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 



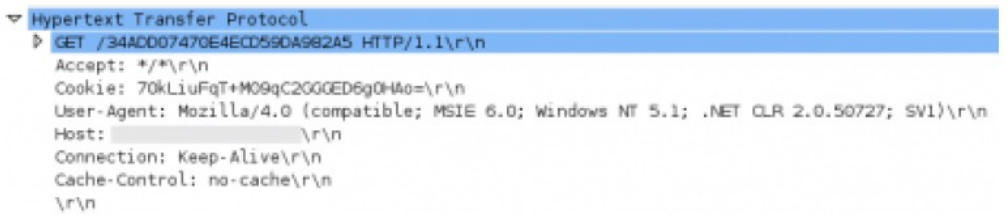
Malware Examples - PlugX 2.0

- Discovered in 2014 (update of the PlugX RAT).
- It uses a custom algorithm for encryption and hides C&C communications in ICMP, HTTP, and DNS 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

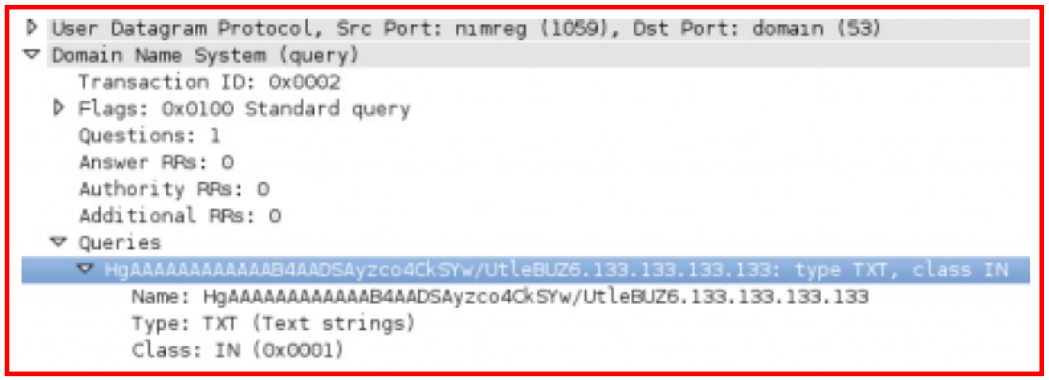
GET (data base64-encoded in Cookie header)



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, HTTP, and DNS 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.

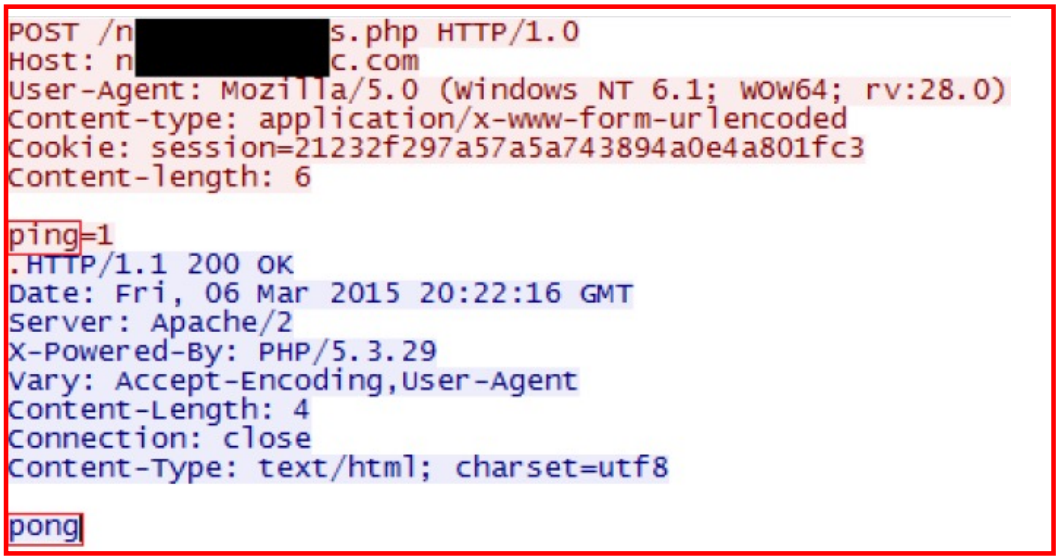


Malware Examples - W32/Foreign.LXES!tr

- 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.

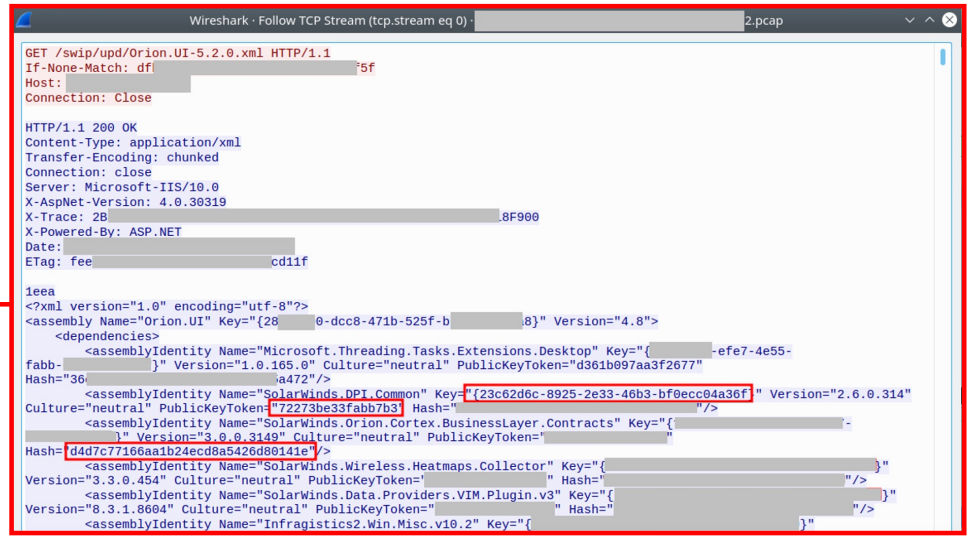
Malware Examples - Sunburst

- Sunburst is a trojanized version of the Orion plugin (Solarwind).
- It targets HTTP traffic.
- Attack Phases:
  - various checks to understand if an analysis tool is running

Malware Examples - Sunburst

- Sunburst is a trojanized version of the Orion plugin (Solarwind).
- It targets HTTP traffic.
- Attack Phases:
  - various checks to understand if an analysis tool is running
  - … (including, opening a backdoor)
  - creates a hidden C&C channel in HTTP.

Sunburst uses HTTP GET or POST requests. The server hides data within HTTP response bodies mimicking benign XML/.NET files. Hidden data is spread across many IDs and strings and extracted via the *\{[0-9a-f-]{36}\}"|"[0-9a-f]{32}"|"[0-9a-f]{16}* regexep.



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.

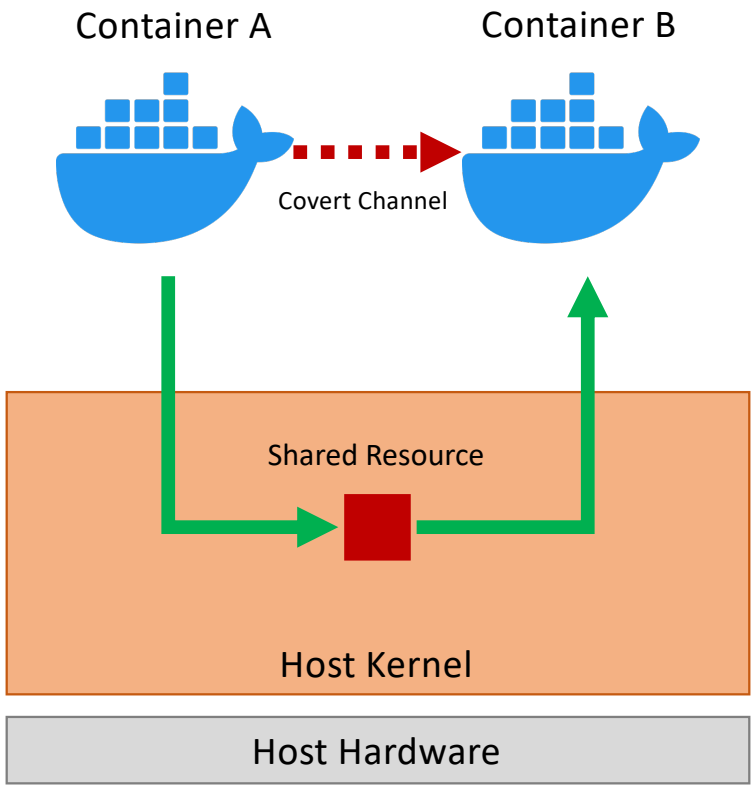
Detection and Mitigation

- 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*.

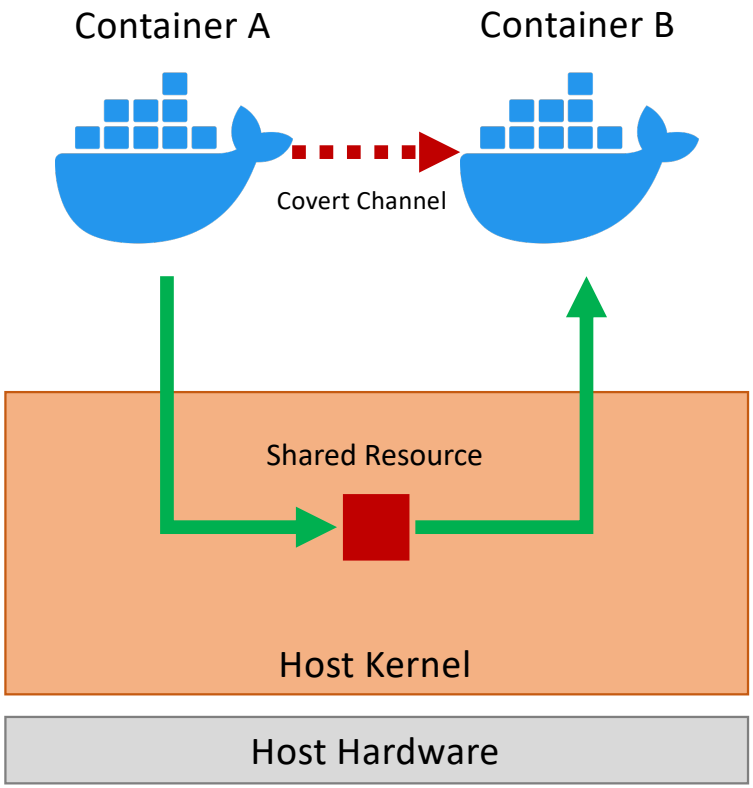
Local Covert Channels

…including the “colluding applications” offensive template

Covert Channels in Docker Containers

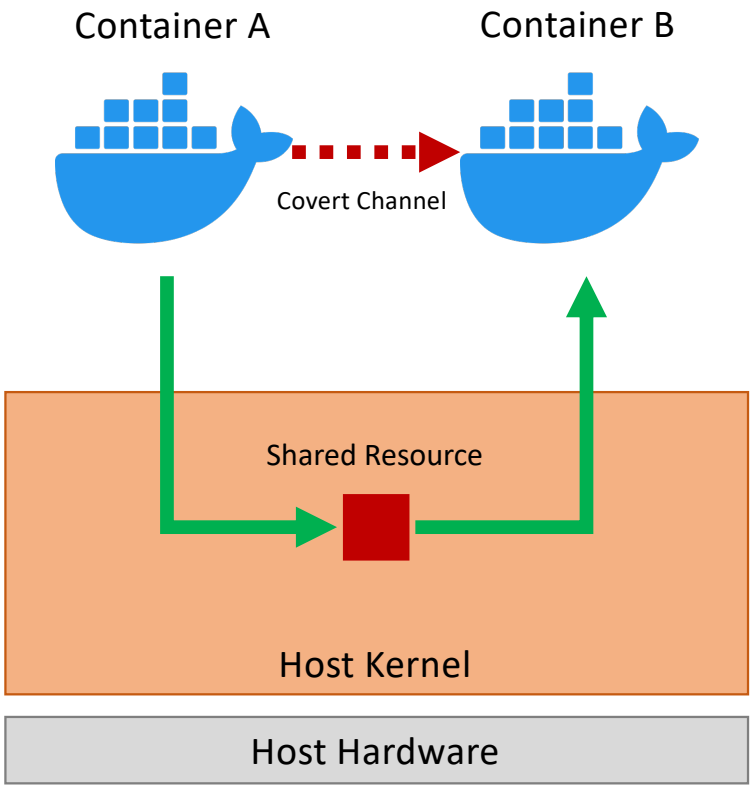


Covert Channels in Docker Containers



- Docker containers can be targeted with a covert channel to leak information or orchestrate attacks.
- 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
  - …

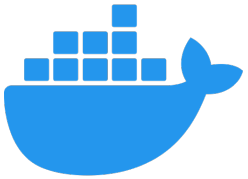
Covert Channels in Docker Containers

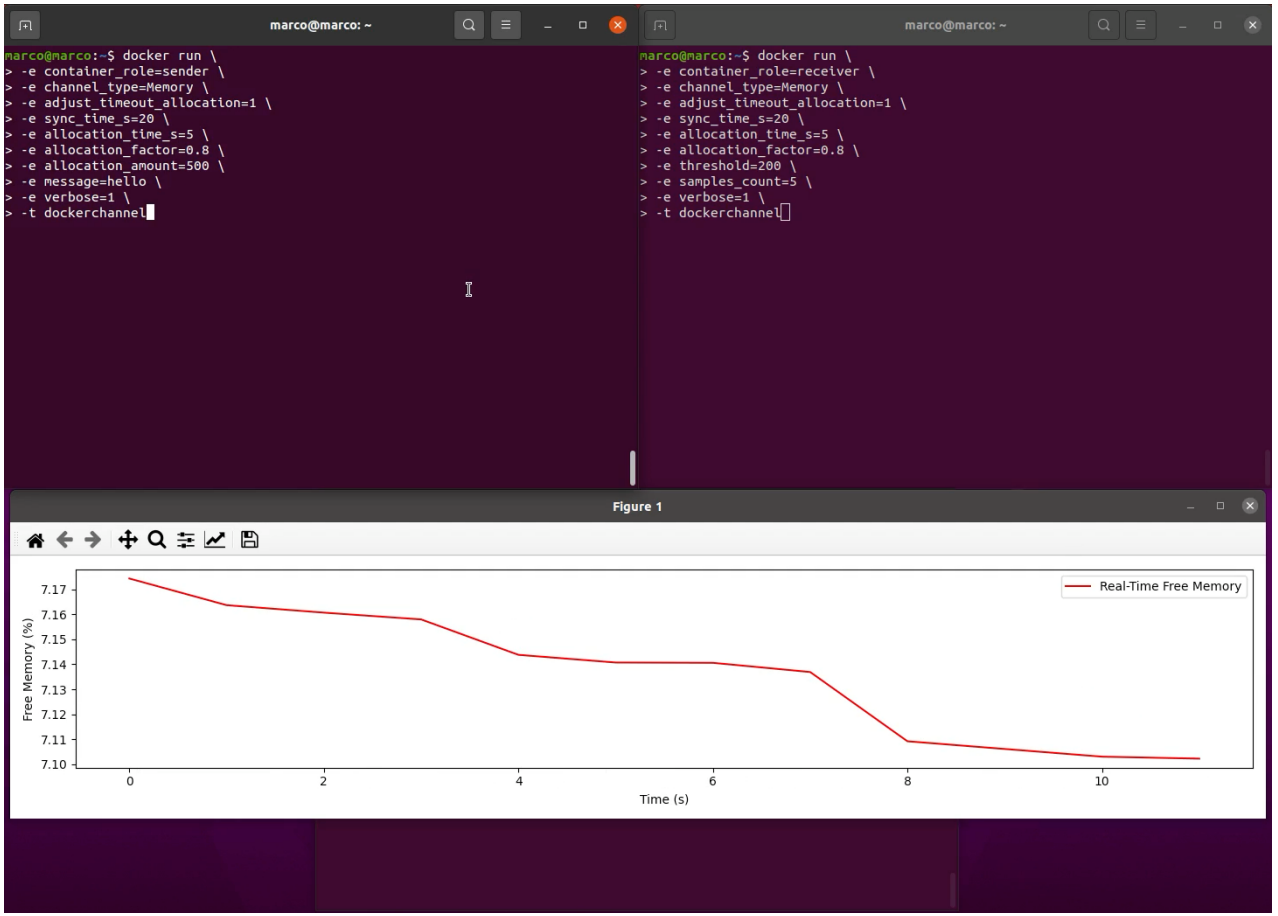


- Example with containers exploiting the free memory available of the host and visible via `/proc/meminfo`.
- **Container A** encodes a binary message by altering the amount of used memory:
  - it allocates 500 MB of memory to encode the bit 1 and sleeps for 5 seconds
  - it sleeps for 5 seconds to encode 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.

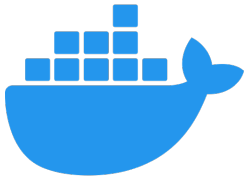
Covert Channels in Docker Containers

Container A





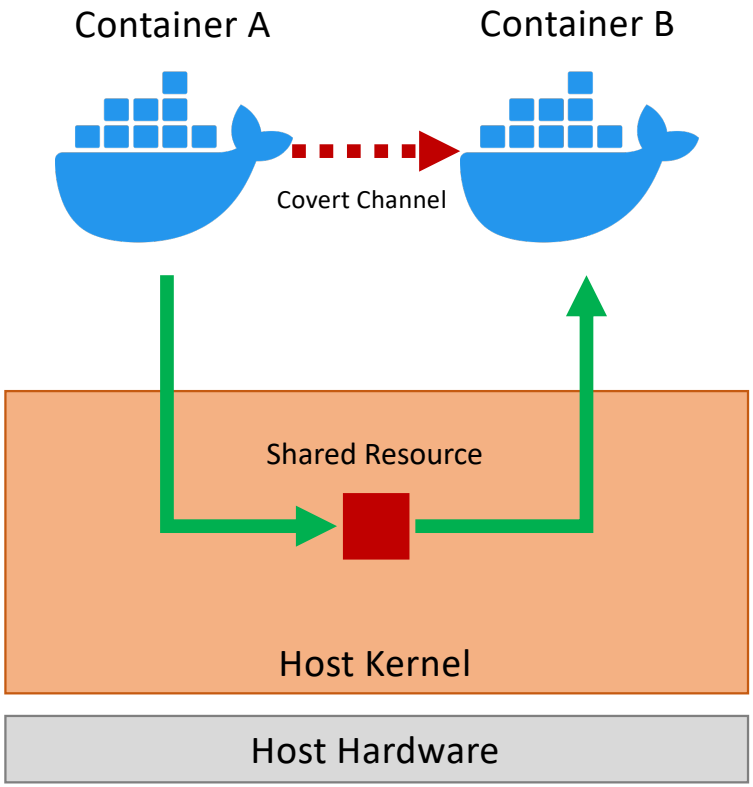
Container B



Shared Resource

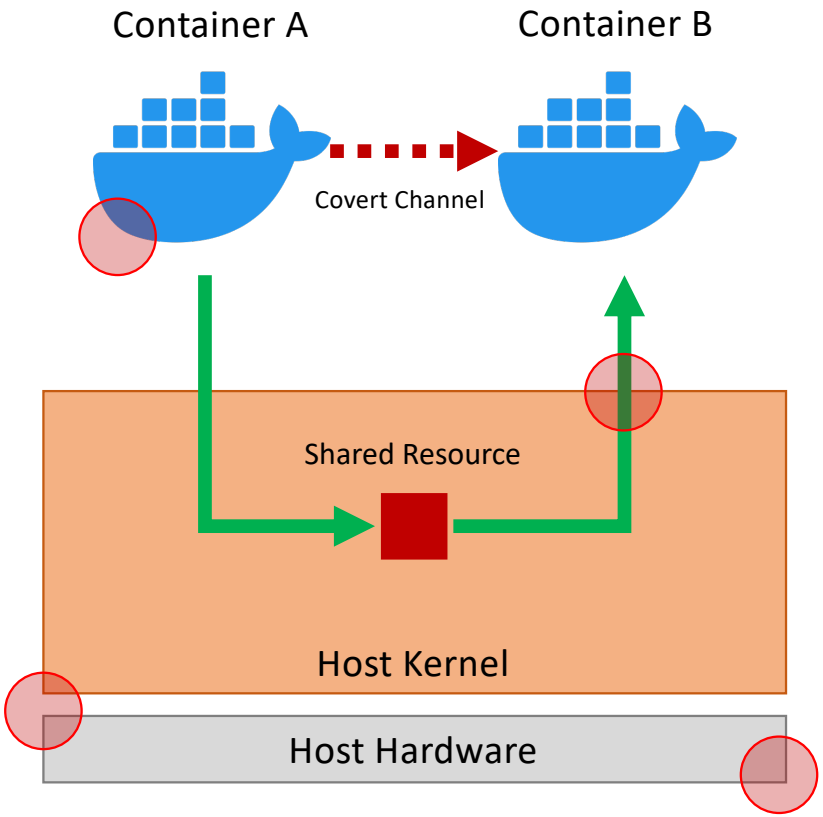
https://github.com/cybersecurity-cnr/DockerChannel

Covert Channels in Docker Containers



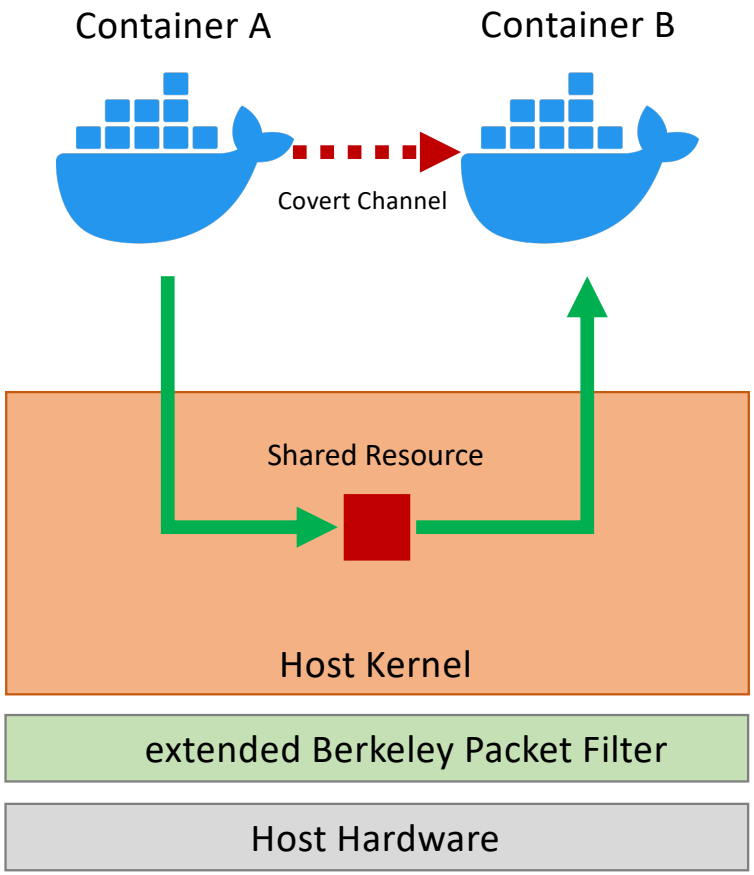
- A possible detection approach is to identify suitable patterns or specific behaviors characterizing the leakage.

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.

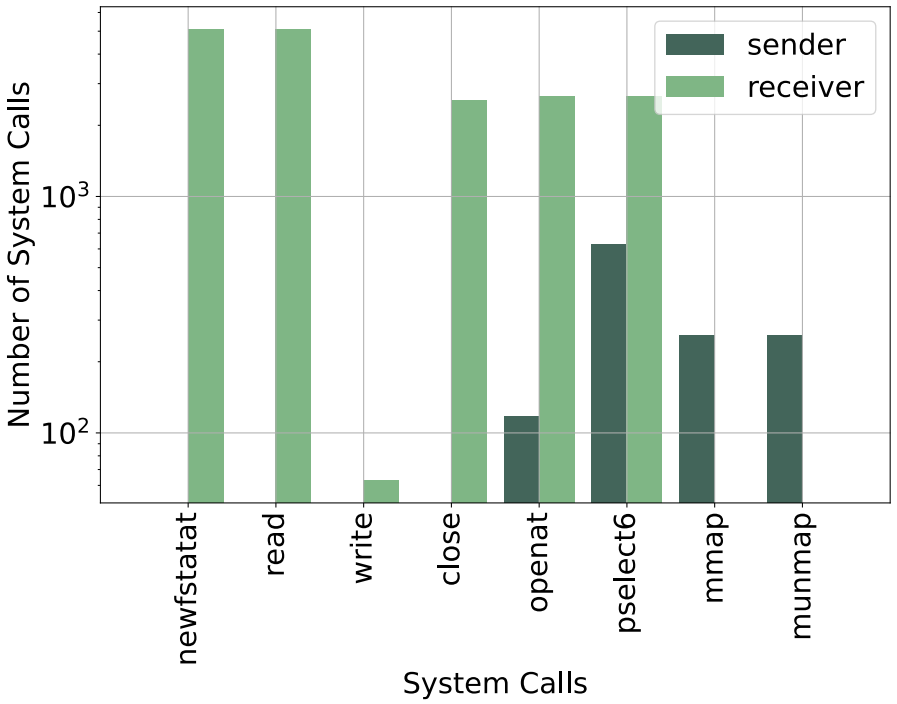
Covert Channels in Docker Containers



- 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

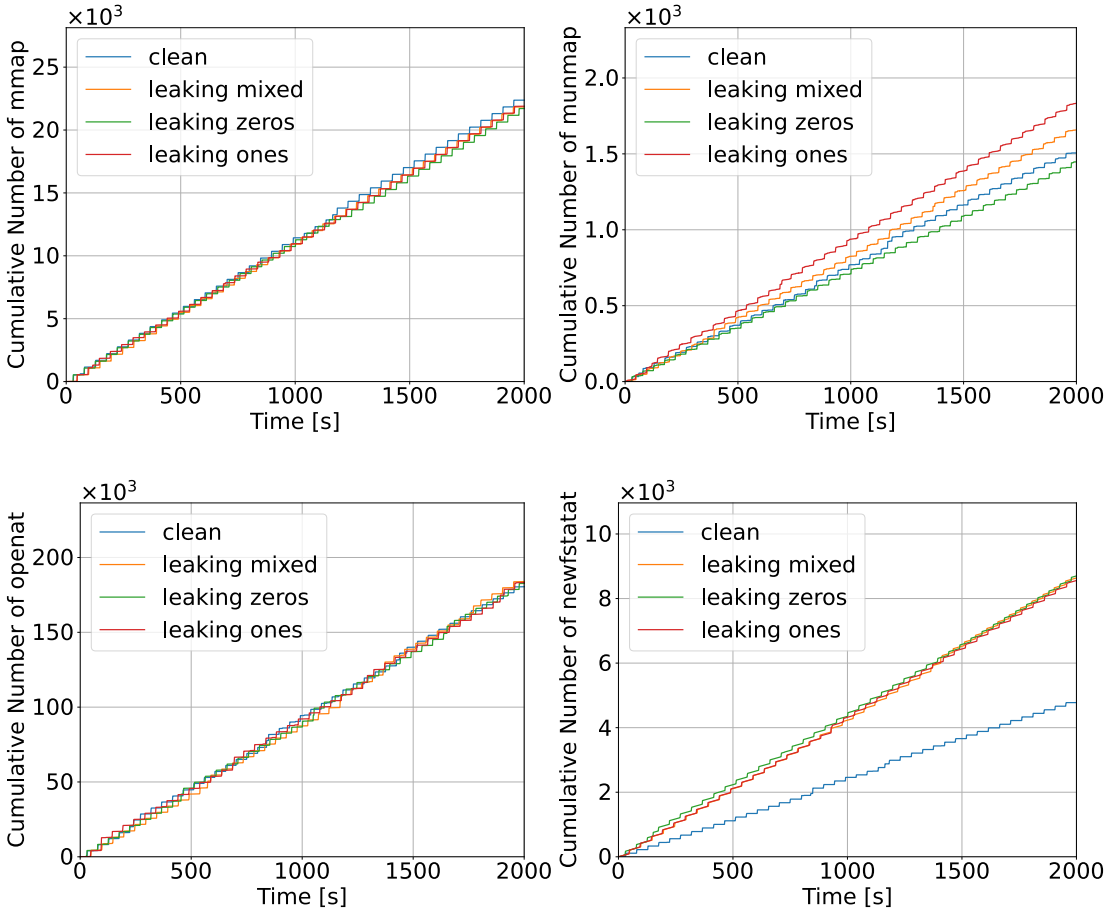
Covert Channels in Docker Containers



- 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.

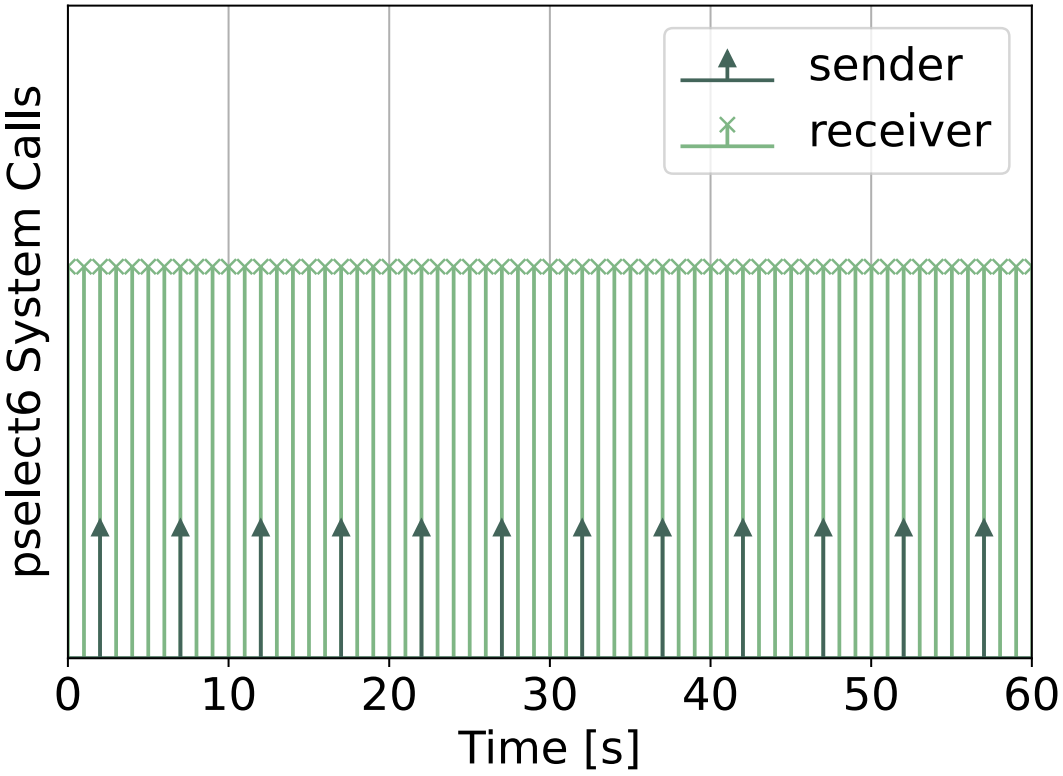
Covert Channels in Docker Containers

- **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.



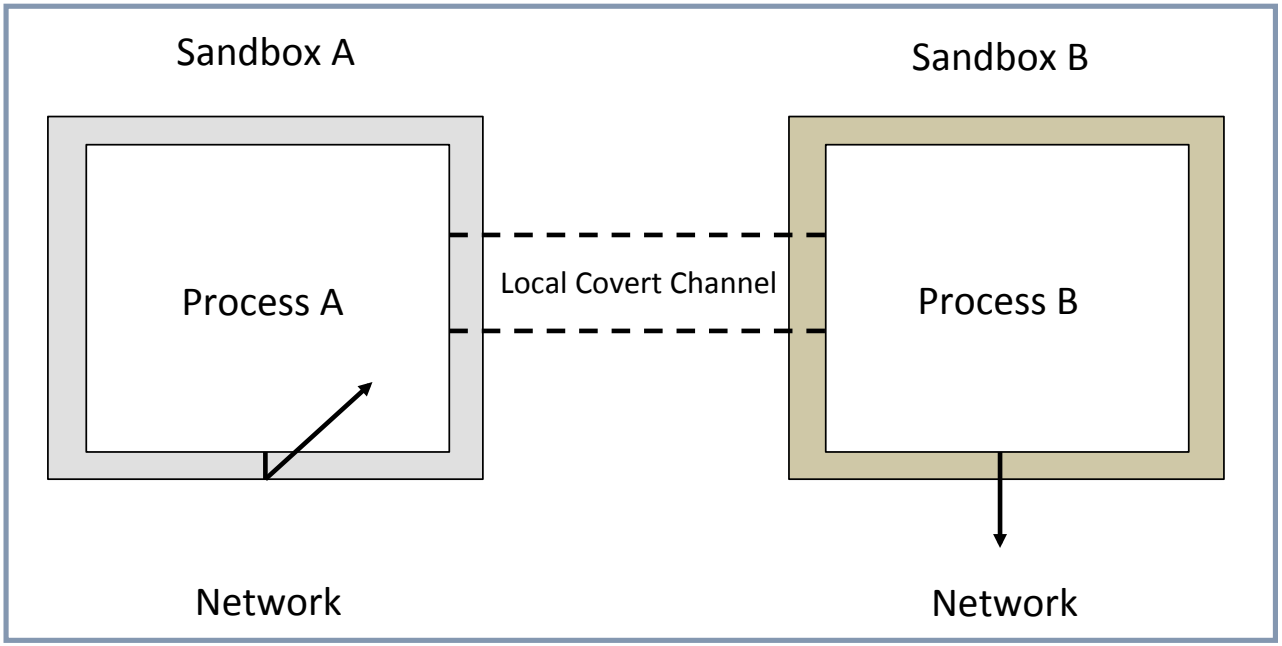
Abstraction Example - Sleeping

- 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.

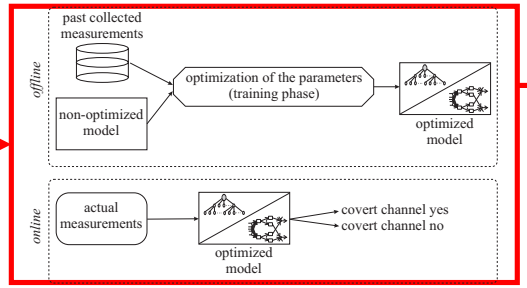


The “Colluding Applications” Scheme

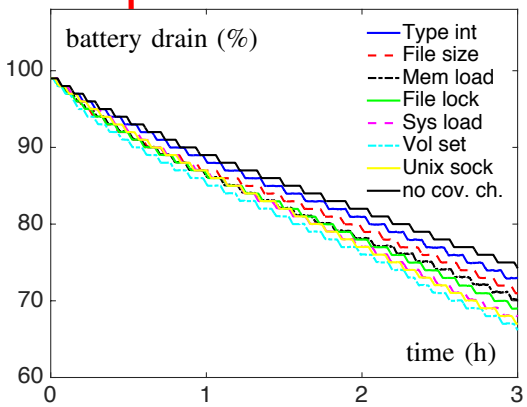
- Covert channels are also at the basis of the “**colluding applications**” offensive template.
- This attack is mainly used in mobile scenarios to **bypass** sandboxing.
- Mobile devices offer many software and hardware features that can act as the carrier:
  - file/socket locks
  - intents
  - volume settings
  - screen lock
  - …
- **Typical scenario**: a process can access personal information, thus the OS prevents the access to the network.
- **Idea**: “leak” information to another process with network privileges.



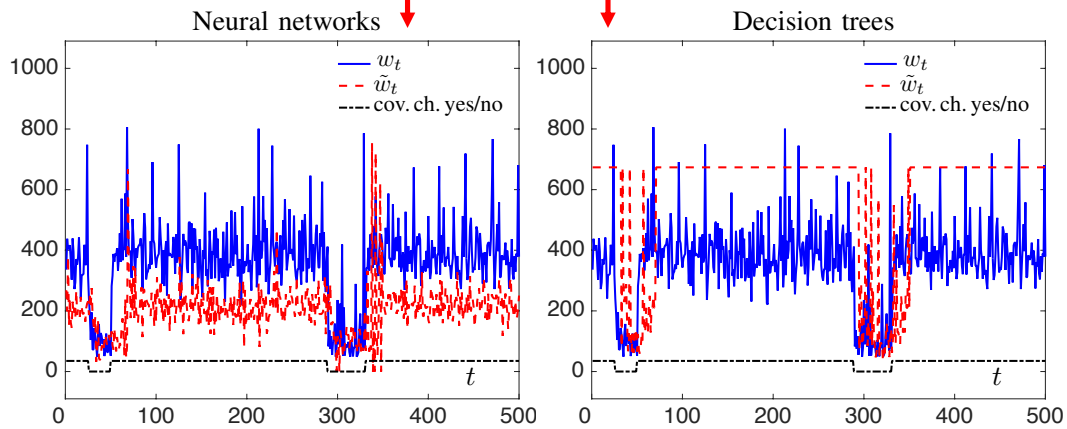
Abstraction Example - Energy



Key idea: develop a database of energy signatures and use artificial intelligence tools (i.e., neural networks and decision trees) to detect applications covertly exchanging data.

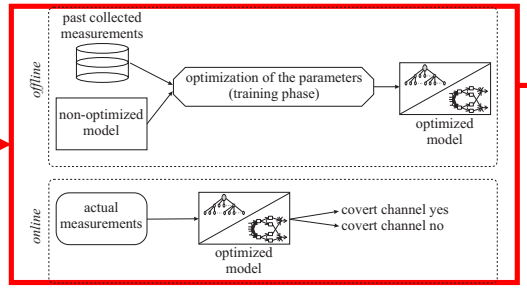


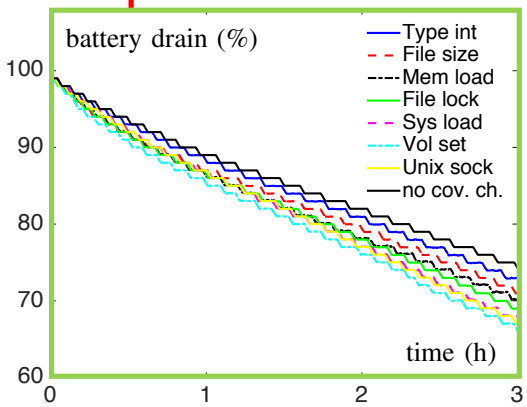
Battery drain for each type of covert channel during 3 hours of repeated experiments.



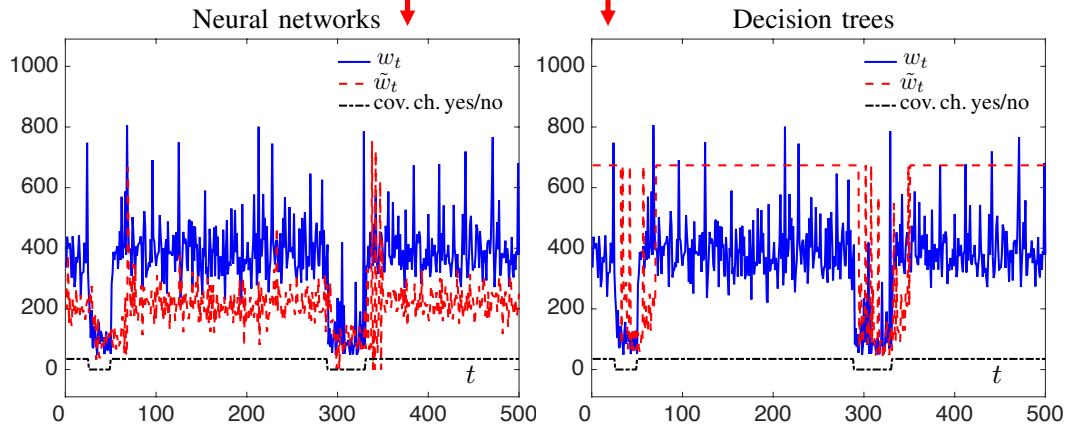
Comparison of real and estimated power consumption of the System process for the Volume Settings covert channel by using the detection method based on neural networks and decision trees.

Abstraction Example - Energy





Battery drain for each type of covert channel during 3 hours of repeated experiments.



Comparison of real and estimated power consumption of the System process for the Volume Settings covert channel by using the detection method based on neural networks and decision trees.

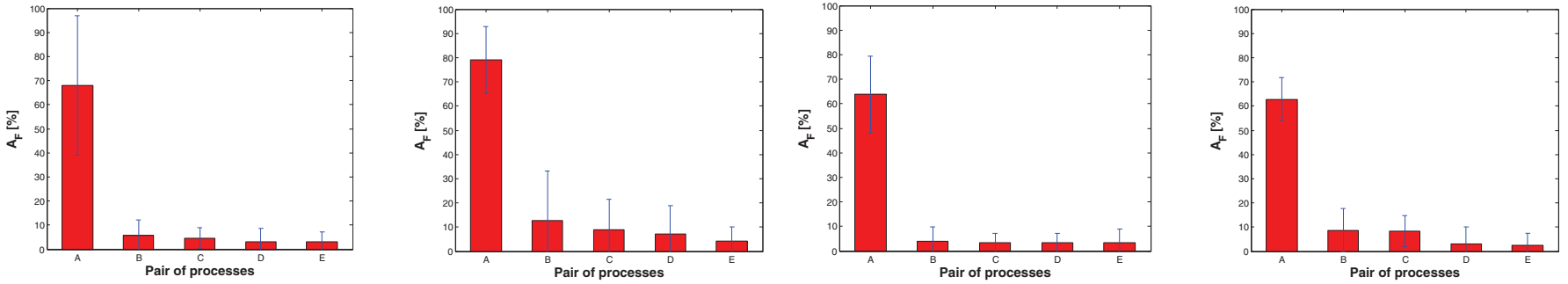
Limit: the battery drain highly depends on how the user interacts with the device and it is typically hard to model!

Abstraction Example - Activity

- Like the case of containers, colluding processes should be active “close” in time.
- **Idea**: mark processes potentially colluding with a **channel-independent metric**. For each process, sample the amount of time spent while:
  - scheduled in user mode (*utime*)
  - scheduled in kernel mode (*stime*)
  - waiting for children in user mode (*cutime*)
  - waiting for children in kernel mode (*cstime*).
- Values collected between **two adjacent** samplings feed a decision rule to **flag** a process.

Abstraction Example - Activity

- Like the case of containers, colluding processes should be active “close” in time.
- **Idea**: mark processes potentially colluding with a **channel-independent metric**. For each process, sample the amount of time spent while:
  - scheduled in user mode (*utime*)
  - scheduled in kernel mode (*stime*)
  - waiting for children in user mode (*cutime*)
  - waiting for children in kernel mode (*cstime*).
- Values collected between **two adjacent** samplings feed a decision rule to **flag** a process.

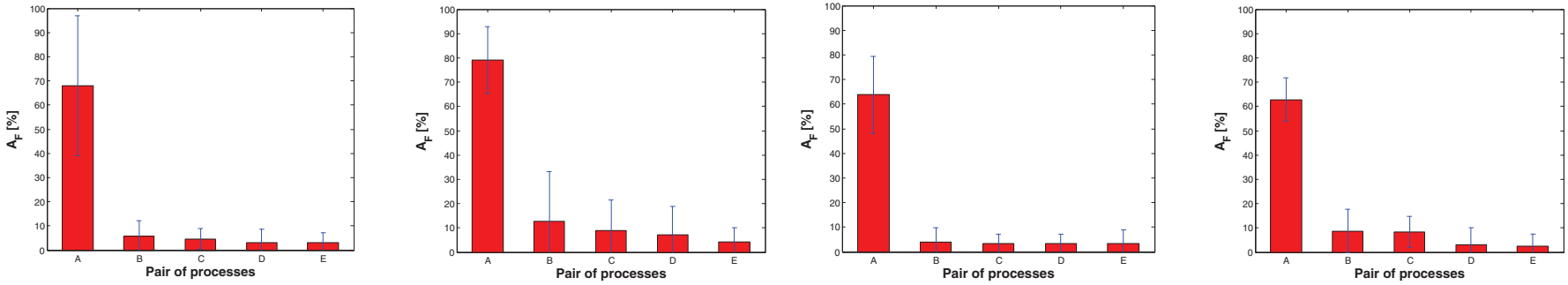


(a) N = 2 (b) N = 5 (c) N = 10 (d) N = 20

Five most active pairs of processes when measuring activity using the sliding window decision rule for the file lock covert channel. N is the length of the window.

Abstraction Example - Activity

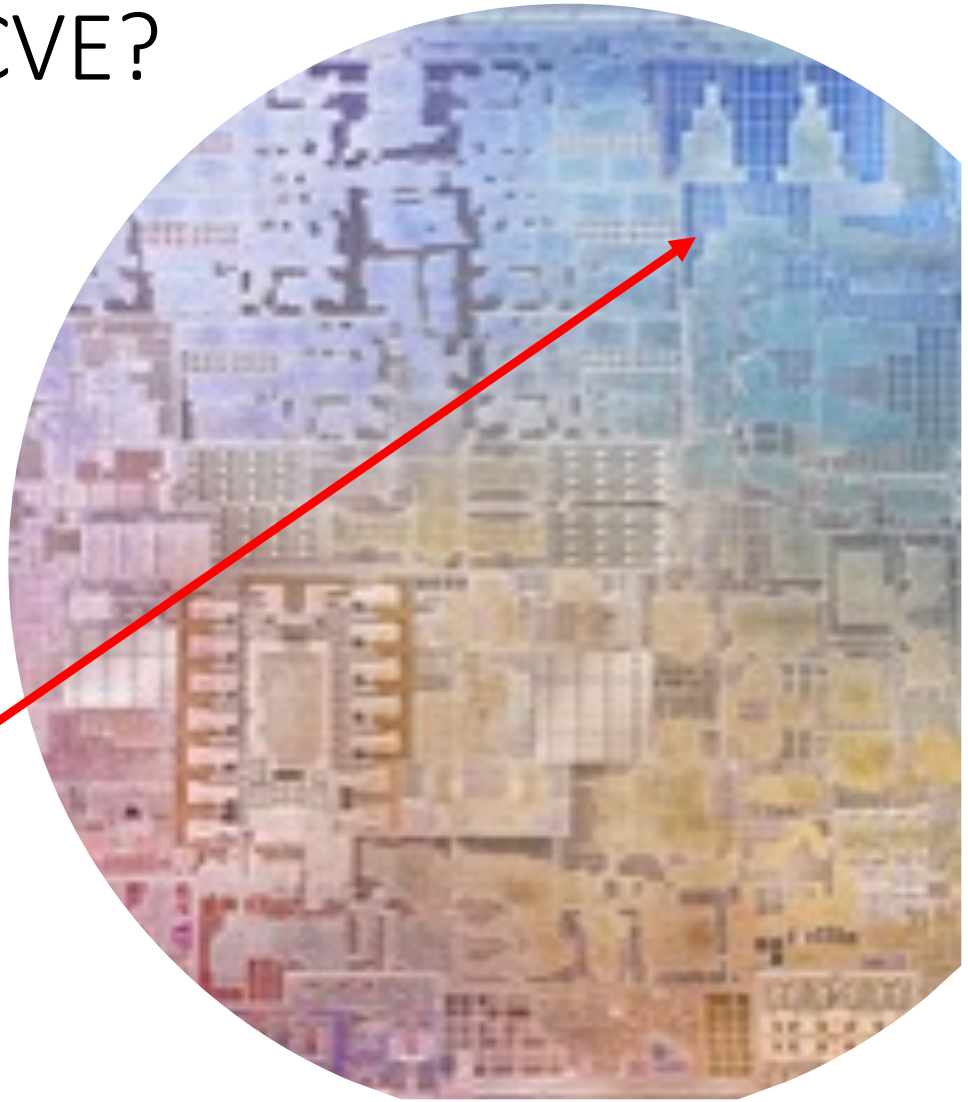
- Like the case of containers, colluding processes should be active “close” in time.
- **Idea**: mark processes potentially colluding with a **channel-independent metric**. For each process, sample the amount of time spent while:
  - scheduled in user mode (*utime*)
  - scheduled in kernel mode (*stime*)
  - waiting for children in user mode (*cutime*)
  - waiting for children in kernel mode (*cstime*).
- Values collected between **two adjacent** samplings feed a decision rule to **flag** a process.



(a) N = 2 (b) N = 5 (c) N = 10 (d) N = 20

The user activity or major “client-server” interactions between processes could void this approach. **A bit of luck**: well-known aggressively-coupled processes, e.g., GUI/HI, can be identified offline to reduce false positives.

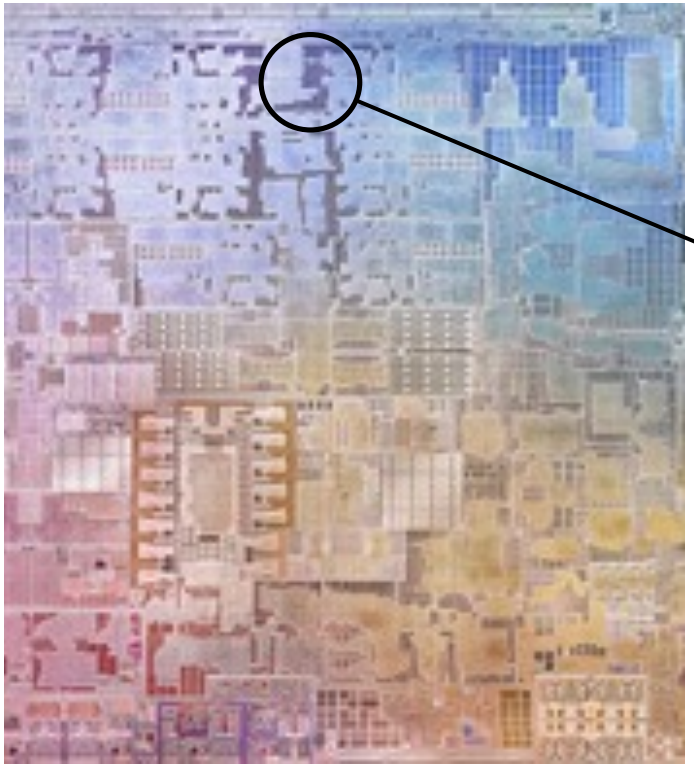
What if a CVE?



M1ssing Register Access Controls Leak EL0 State
(CVE-2021-30747)
https://m1racles.com

Not a “serious threat”, rather it is a privacy leak

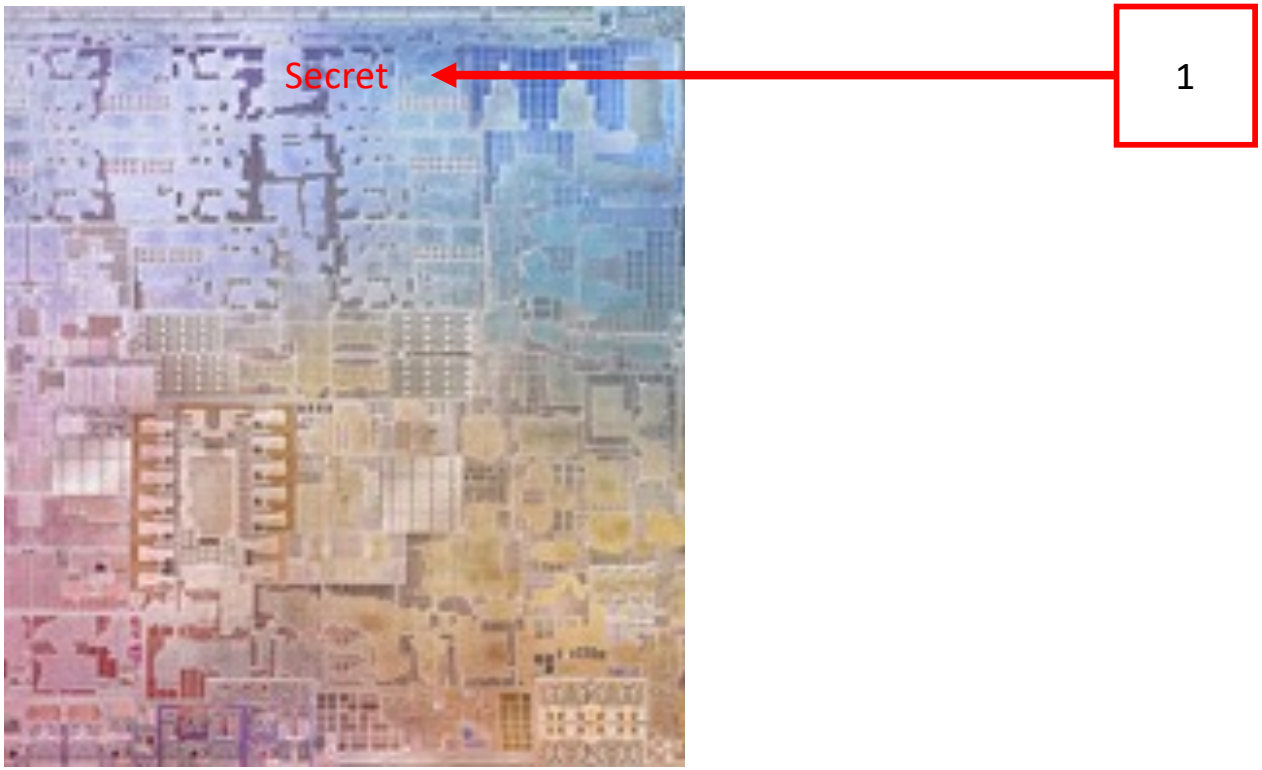
Covert Channels Through M1RACLES



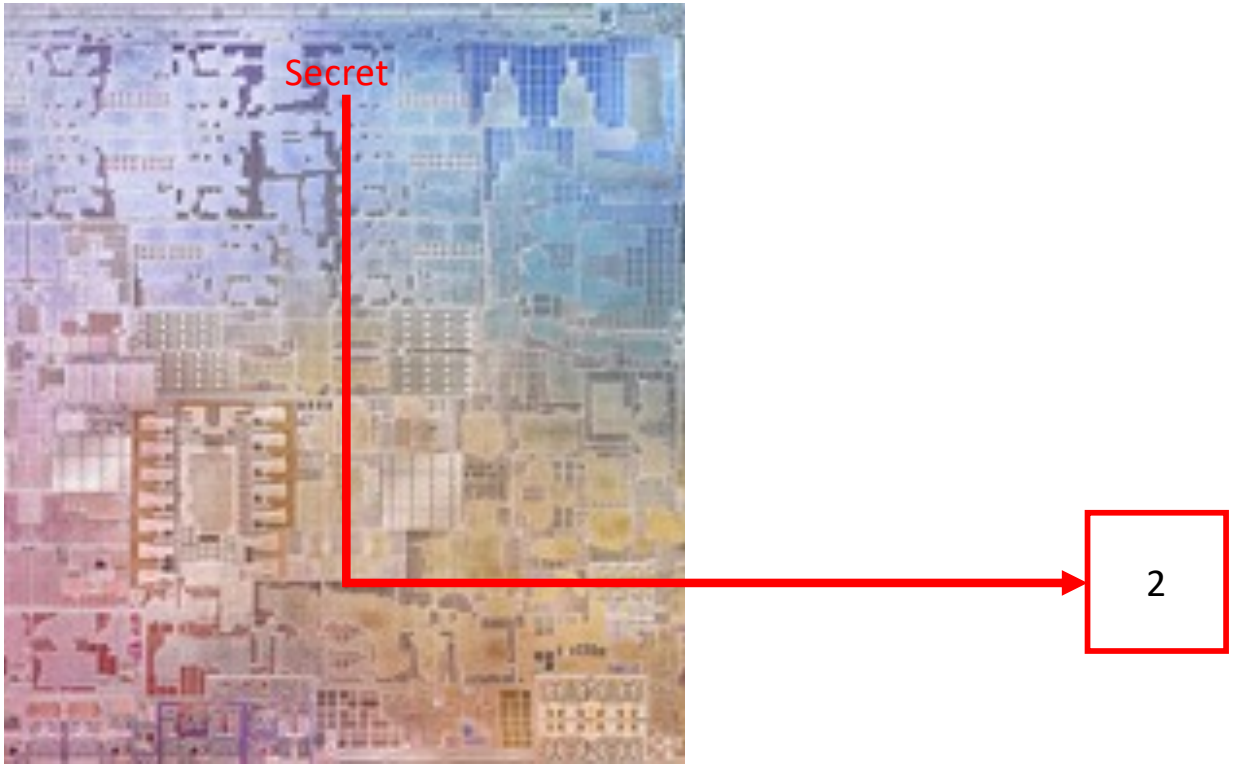
ARM System Register: its purpose is unknown, but probably not intended to make it accessible to EL0.

EL0 (Exception Level) User Applications

Covert Channels Through M1RACLES



Covert Channels Through M1RACLES



Emerging Usages of Covert Channels

…including the ubiquitous application to AI

Software Obfuscation

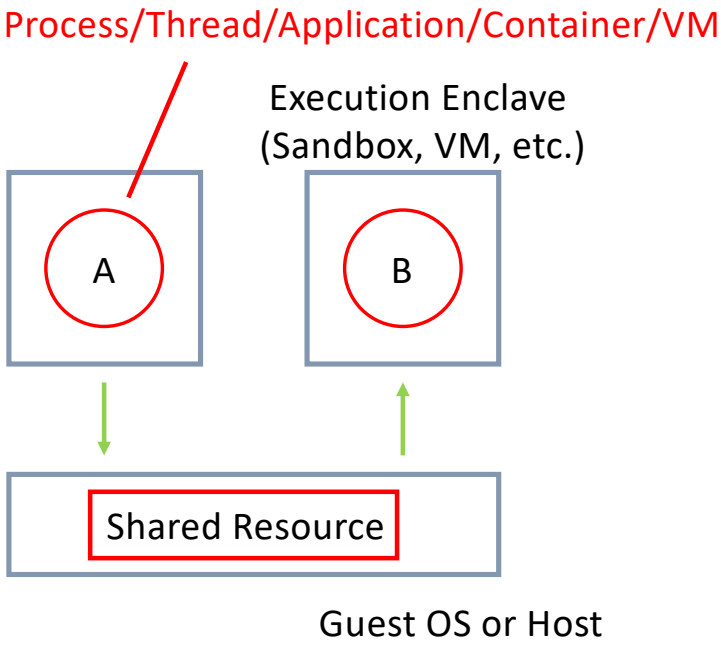
- Covert channels are a **double-edged sword**.
- A major recent example is software protection, e.g., to make reverse engineering attempts harder:
  - IP enforcement
  - resist against forensics and threat analysis.
- The typical approach is to use **code obfuscation**.
- The “Colluding Applications” scheme can be used to implement timing or probabilistic **covert channels** for obfuscating the code or enforcing specific execution behaviors.

```
void P() {
    int a = b;
}
```

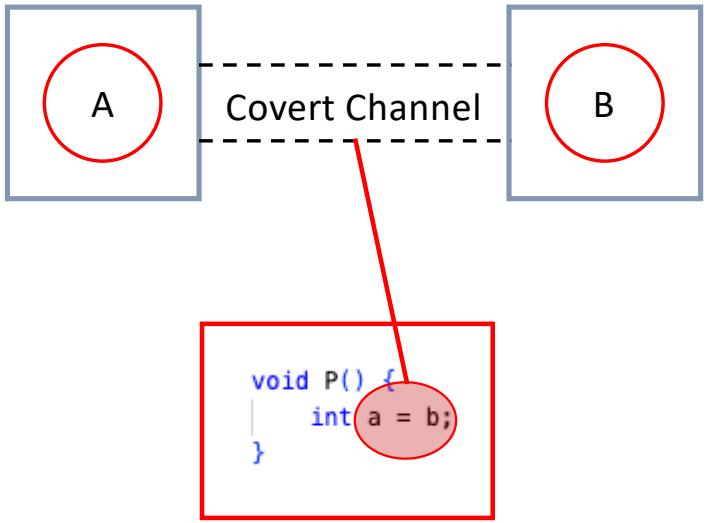
```
void P() {
    int i,a = 0;
    for(i=0;i<b;i++) {
        a++;
    }
}
```

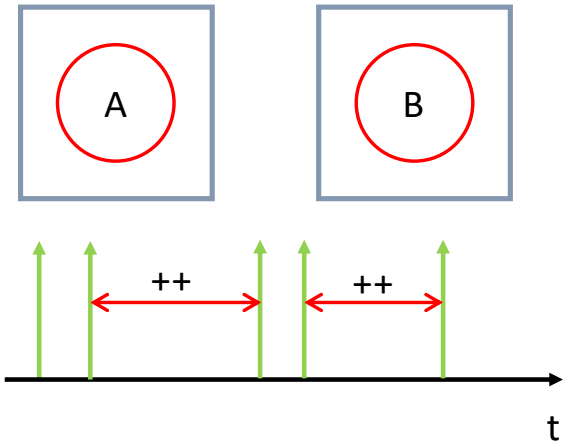
J. Stephens, et al., "Probabilistic Obfuscation Through Covert Channels", *European Symposium on Security and Privacy,* 2018, pp. 243-257

Software Obfuscation

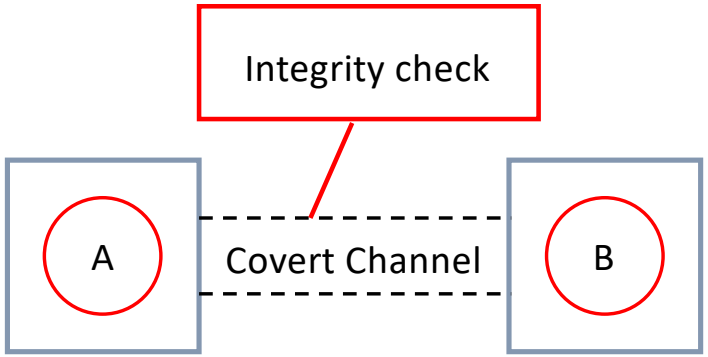


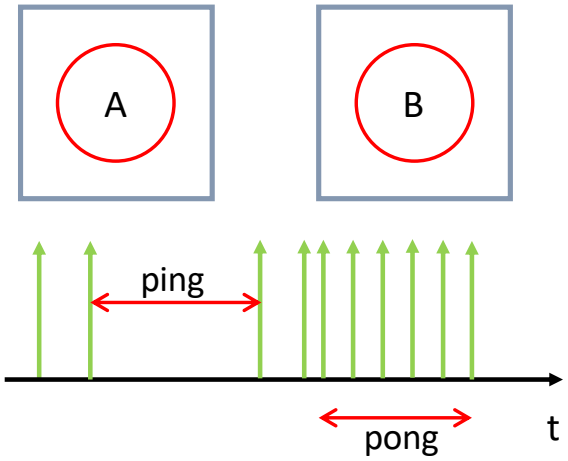
Software Obfuscation





Software Obfuscation





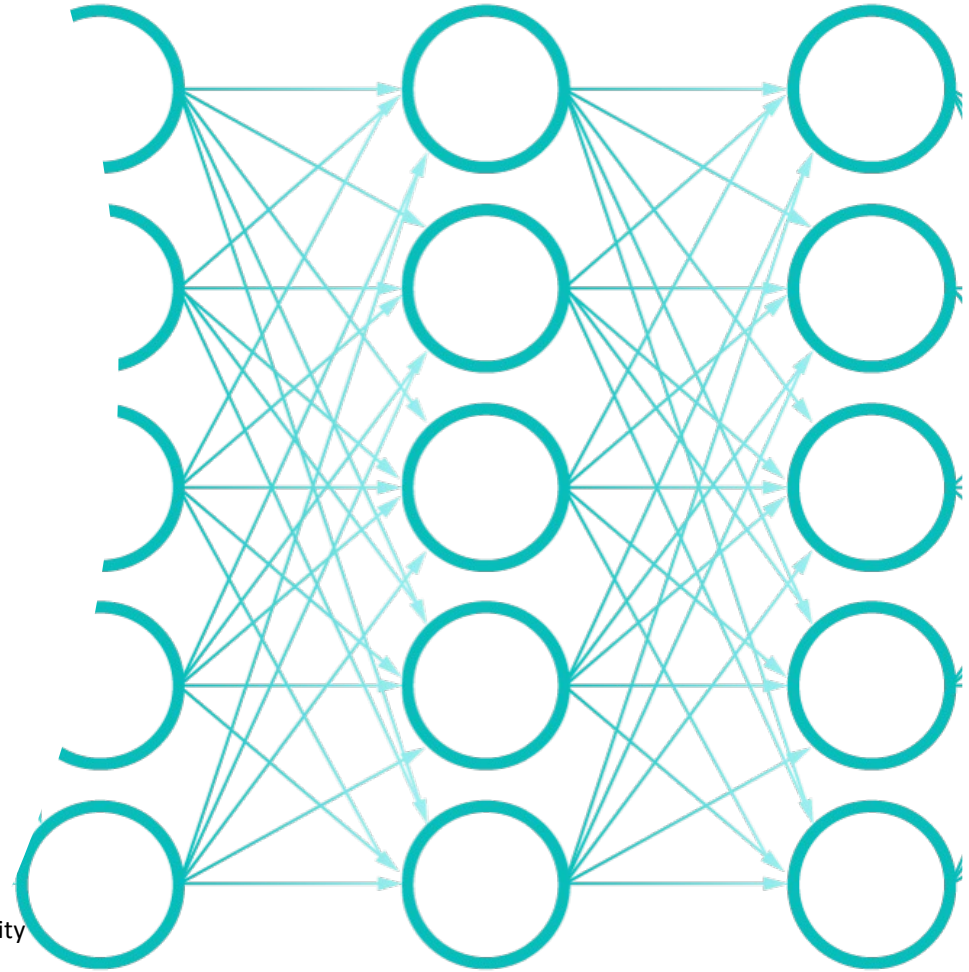
AI and Covert Communications

- In 2021, the industry of machine learning had a value of **15.5$ billions**:
  - AI-as-a-Service and ML-as-a-Service
  - IP protection
  - tracking unfair usages.
- Covert channels and information hiding techniques surely **can play a role**.
- Diffusion of AI makes it an attractive **carrier** for **attackers**.

https://blogs.sap.com/2021/11/08/protect-your-machine-learning-models-with-watermarking/

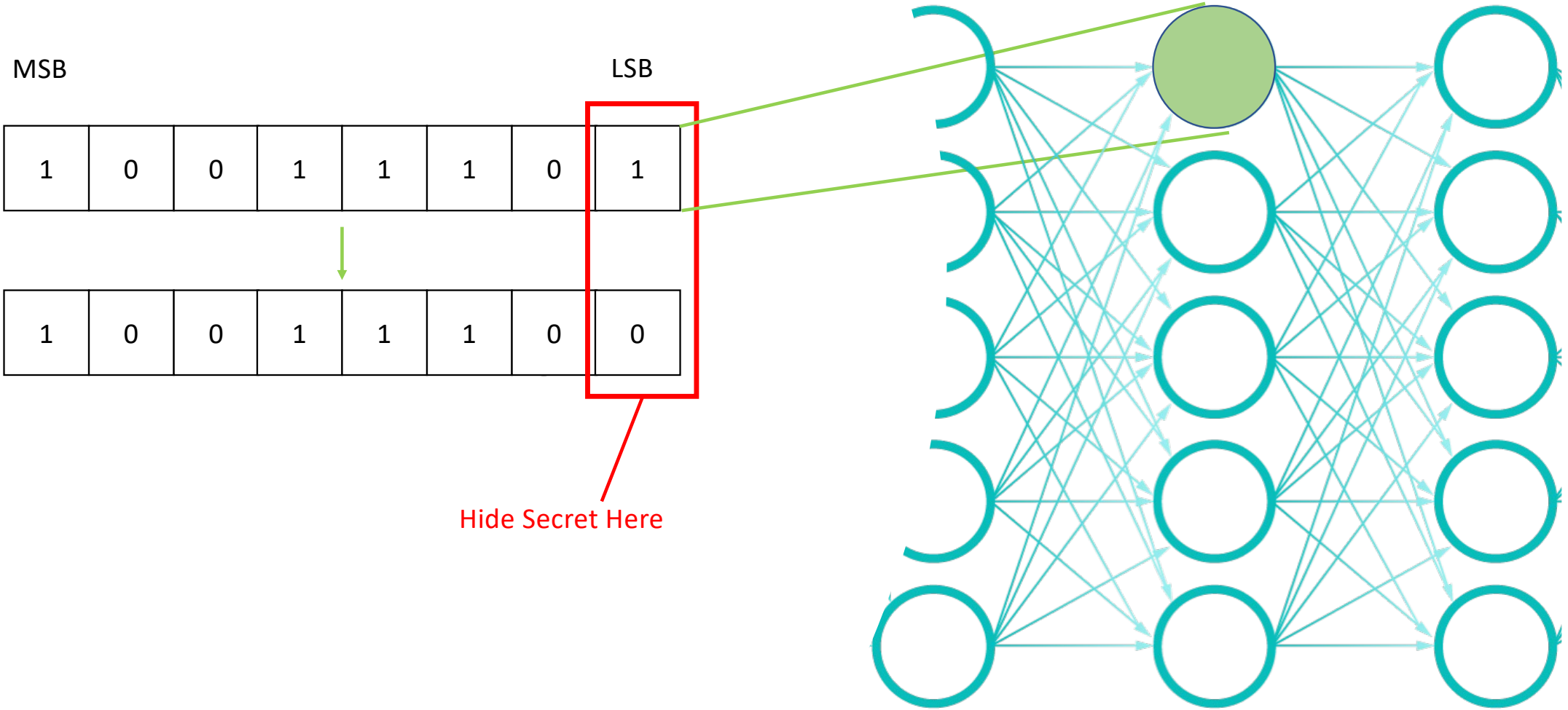
AI and Covert Communications

- A covert communication can be set via:
  - classical **LSB approaches** (i.e., the last bit(s) of some model parameters are altered to encode arbitrary data)
  - exploiting **resilience** of deep neural networks to introduce internal errors by overwriting the model with secret data and do not re-train the resulting “broken neurons”
  - **mapping** techniques, i.e., arbitrary information is placed by altering parameters with similar or closest values.

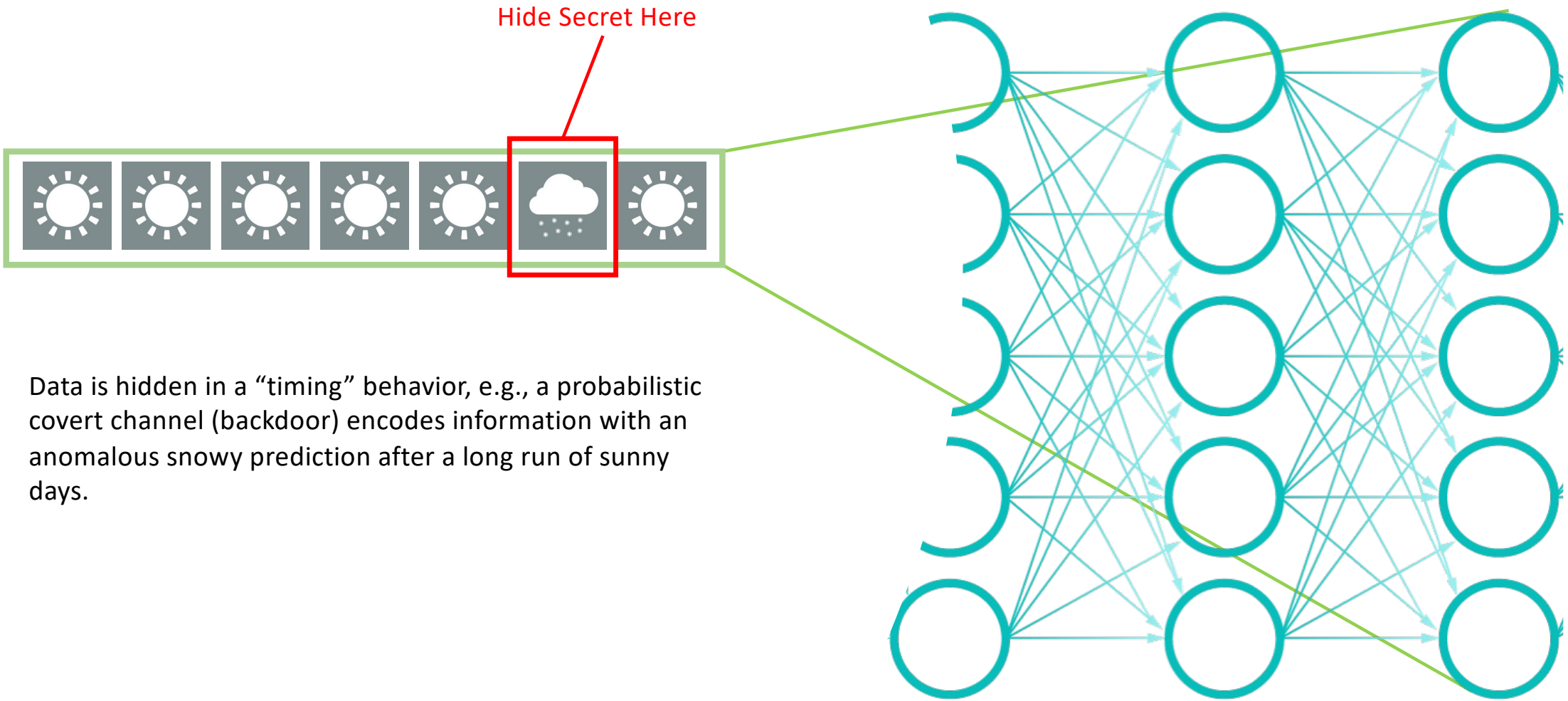


T. Liu, et al., "Stegonet: Turn Deep Neural Network into a Stegomalware”, Proc. of the Annual Computer Security Applications Conference, New York, NY, USA, 2020, p. 928–938.

AI and Covert Communications



AI and Covert Communications



Data is hidden in a “timing” behavior, e.g., a probabilistic covert channel (backdoor) encodes information with an anomalous snowy prediction after a long run of sunny days.

Research Challenges

Research Challenges

Research Challenges

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

Research 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

Research Challenges

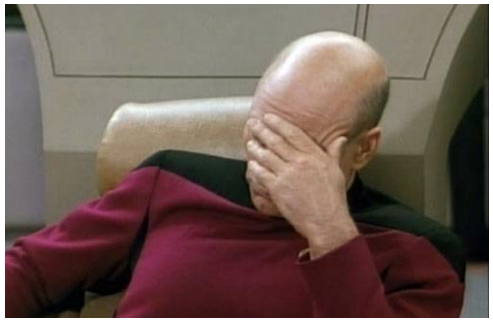
- There is **not** a **unique** hiding strategy and there is **not** a **general** defensive methodology.
- **Anticipating** ambiguities rather than fixing them later:
  - this requires formalization, by-design approaches, and SBoM-like tests
  - bring back hiding mechanisms to recurrent **patterns**
  - complex interplays difficult to capture “by hand”.
- Make existent approaches scalable:
  - especially for network traffic and Internet-scale scenarios
  - 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 when addressing covert channels:
  - carrier is **not** known *a priori* (e.g., network traffic, system-wide behaviors, and AI)
  - **heterogenous** set of protocols, software objects and data types
  - mixed **techniques** (storage or timing)
  - **GDPR-like** constraints
  - **scalability**
  - …
- Detection and mitigation are:
  - method-dependent
  - poorly generalizable.
- We should aim at **preventing** and **eliminating** ambiguities.



Collaborations and Research Projects

- Joint research between UniPD and CNR-IMATI aims at evaluating covert communications in **containerized** environments to:
  - outline **new** attack templates
  - develop **defensive** mechanisms
  - pursue **elimination** and **detection.**
- Possible joint research topics or projects:
  - “force” **schedulers** used to orchestrate containers for creating covert communications
  - exchange of hidden information to guarantee the **integrity** of microservices or to check whether a container is used **outside** its original deployment
  - eBPF for **anomaly detection** and **neutrality checking** tasks.

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.

L. Caviglione, M. Gaggero, J.-F. Lalande, W. Mazurczyk, M. Urbański, “Seeing the Unseen: Revealing Mobile Malware Hidden Communications via Energy Consumption and Artificial Intelligence”, IEEE Transactions on Information Forensics and Security, Vol. 11, No. 4, pp. 799 - 810, April 2016.

M. Urbański, W. Mazurczyk, J.-F. Lalande, **L. Caviglione**, “Detecting Local Covert Channels Using Process Activity Correlation on Android Smartphones”, International Journal of Computer Systems Science and Engineering, Vol. 32, No. 2, pp. 71 - 80, March 2017.

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.

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.

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

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.

L. Caviglione, C. Comito, M. Guarascio, G. Manco, “Emerging Challenges and Perspectives in Deep Learning Model Security: A Brief Survey”, Systems and Soft Computing, Vol. 5, pp. 1 - 7, December 2023.

Thank You!
Questions?

@lucacaviglione