MARIST

School of Computer Science and Mathematics

Covert Channels and Injection Vulnerabilities in IEEE Precision Time Protocol

Casimer DeCusatis, Ph.D. & IBM D.E. Emeritus, Elizabeth Herrera and Luke Jacobs Marist College

elizabeth.herrera1@marist.edu

luke.jacobs1@marist.edu

Paul Wojciak, D.E., Clay Kaiser, and Steve Guendert, IBM

wojciak@us.ibm.com

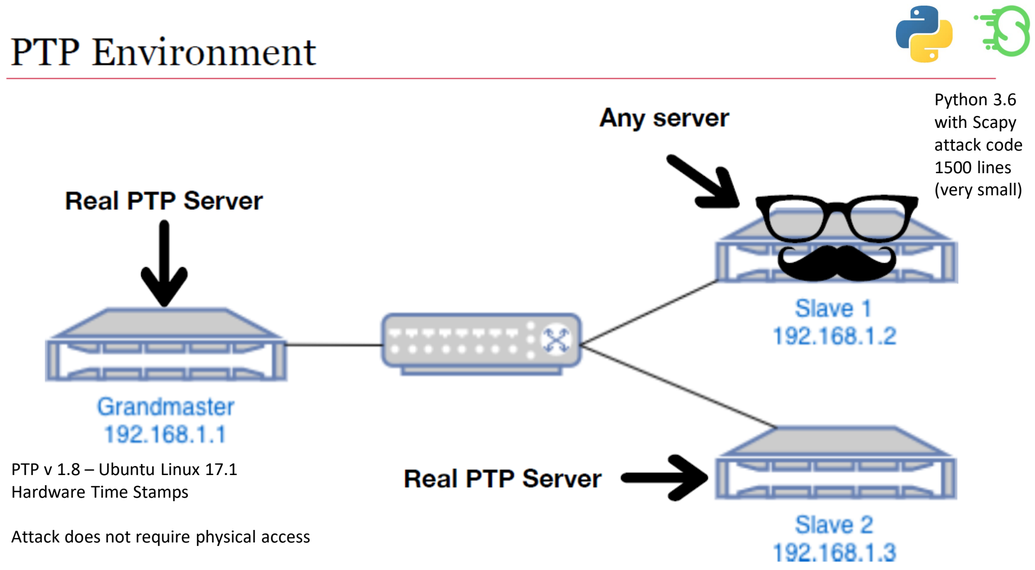
IBM

Overview – What is PTP?

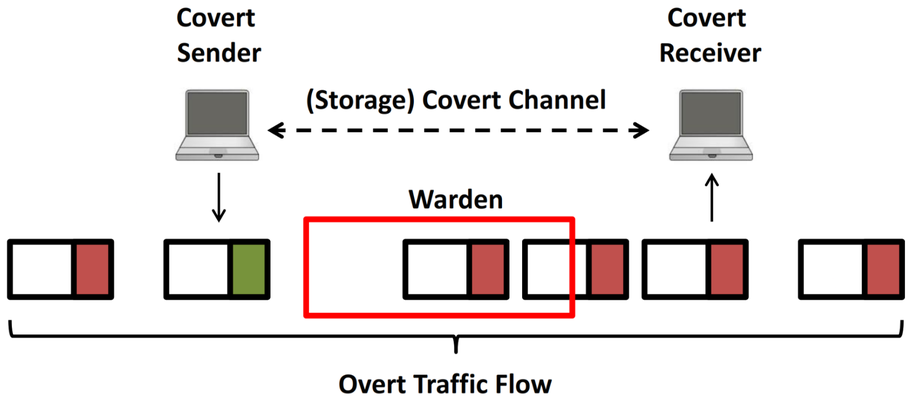
- The IEEE 1588 standard Precision Time Protocol standard (PTP) is a follow-on to the well known Network Time Protocol (NTP) which provides highly accurate (nanosecond or better) synchronized data center clock signals.

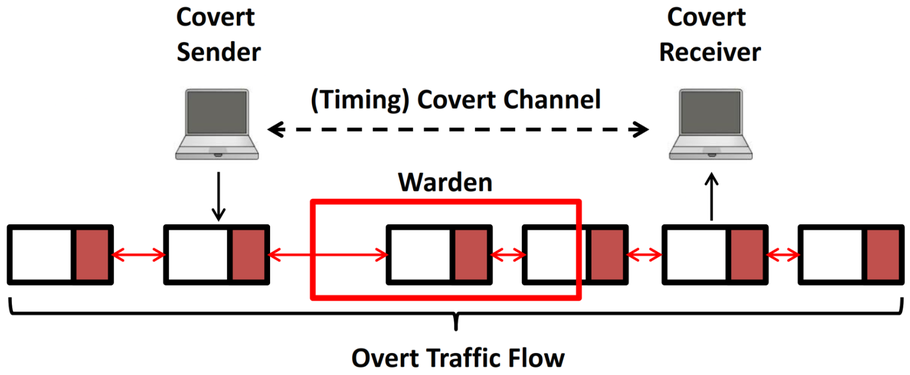
- Cyberattacks which destroy clock synchronization have devastating consequences.
  - Does not preserve order of transactions; critical issue for IBM Z Systems and Next Generation GDPS
  - Impacts event scheduling (backup/recovery with incorrect timestamps) including recovery time point/objective, causality violation
  - Induce time skips, temporal vortex, or complete loss of clock synchronization to all clients

- Timing channels can also be co-opted to infiltrate/exfiltrate data even if timing network is not connected to the Internet



Covert Channels & Vulnerabilities





- **Covert channels** transfer information between processes that are not normally allowed to communicate based on cybersecurity policy
- Ideally the communication is difficult to detect by other processes unless all meta-data fields are validated, and does not obviously impede normal operation
- Covert channels were not designed for communication, and therefore often exhibit low data rates, lack of redundancy/retransmission or error correction capability
- Often used for data exfiltration or to install/update malware
- Prior documented examples include DNS, NTP, and others (see N. Tsapakis, Virusbulletin.com, April 2019)

- **Vulnerabilities** may exist if metadata such as a packet header field is not validated, making the field vulnerable to different types of data injection attacks

PTP Packet Headers as Covert Channels

```
Precision Time Protocol (IEEE1588)
  0000 .... = transportSpecific: 0x00
     ...0 .... = V1 Compatibility: False
  .... 0001 = messageId: Delay_Req Message (0x01)
  .... 0010 = versionPTP: 2
  messageLength: 44
  subdomainNumber: 0
  flags: 0x0000
     0... .... .... .... = PTP_SECURITY: False
     .0.. .... .... .... = PTP profile Specific 2: False
     ..0. .... .... .... = PTP profile Specific 1: False
     .... .0.. .... .... = PTP_UNICAST: False
     .... ..0. .... .... = PTP_TWO_STEP: False
     .... ...0 .... .... = PTP_ALTERNATE_MASTER: False
     .... .... ..0. .... = FREQUENCY_TRACEABLE: False
     .... .... ...0 .... = TIME_TRACEABLE: False
     .... .... .... 0... = PTP_TIMESCALE: False
     .... .... .... .0.. = PTP_UTC_REASONABLE: False
     .... .... .... ..0. = PTP_LI_59: False
     .... .... .... ...0 = PTP_LI_61: False
  correction: 59345.000000 nanoseconds
     correction: Ns: 59345 nanoseconds
     correctionSubNs: 0.000000 nanoseconds
  clockIdentity: 0x001d9cfffeb1acfe
  sourcePortID: 1
  sequenceId: 15638
  control: Delay_Req Message (1)
  logMessagePeriod: 127
  originTimestamp (seconds): 1436270274
  originTimestamp (nanoseconds): 26902220
```

1. Sniff for incoming packets to determine the next sequence ID (only to avoid packet collision).
2. Construct spoofed packets, for example
   - 8 bytes is inserted into the correction field during packet creation.
   - 8 bytes can also optionally be inserted into the clock identity field.
3. Read hexadecimal data from a text file to simulate data exfiltration.
4. Send spoofed packet to source node.
5. Send packets in time intervals that mimic normal occurrences.

   Undetectable for certain packets, such as delay_request messages

Covert Channel Summary

- Covert Communication Channels for PTP demonstrated experimentally
  - Correction Field as used in Delay_request field and Sync Followup field (non-colliding sequence IDs, i.e. enterprise profile)
  - Reserved Field
  - Detectable exfiltration in three other channels

CCAC MARIST

Header Field (Announce Packets)	Data Exfiltration / Covert Channel	Data Injection Attack
Grand Master Clock ID	**Exfiltration possible, not covert**	Intermittent Temporal Vortex
Grand Master Clock Accuracy	**Exfiltration possible, not covert**	Intermittent Temporal Vortex
Origin Timestamp Sec and Origin Timestamp Nanosec	**Exfiltration possible, not covert**	Intermittent Temporal Vortex
Reserved Field	Covert Channel	**None**
Correction Field (also for Sync Packets and Delay_Request Messages)	Covert Channel	MITM Attack Clock Frequency Attack

Announce DoS – spam announce packets at the follower

Announce DoS

192.168.1.1	224.0.1.129	PTPv2	63854	106	Announce Message
192.168.1.1	224.0.1.129	PTPv2	55201	106	Announce Message
192.168.1.1	224.0.1.129	PTPv2	55200	106	Announce Message
192.168.1.1	224.0.1.129	PTPv2	55199	106	Announce Message
192.168.1.1	224.0.1.129	PTPv2	55198	106	Announce Message
192.168.1.1	224.0.1.129	PTPv2	55197	106	Announce Message
192.168.1.1	224.0.1.129	PTPv2	55196	106	Announce Message
192.168.1.3	224.0.1.129	PTPv2	3177	86	Delay_Req Message
192.168.1.1	224.0.1.129	PTPv2	55195	106	Announce Message
192.168.1.1	224.0.1.129	PTPv2	55194	106	Announce Message
192.168.1.1	224.0.1.129	PTPv2	55193	106	Announce Message
192.168.1.1	224.0.1.129	PTPv2	55051	106	Announce Message
192.168.1.1	224.0.1.129	PTPv2	55050	106	Announce Message

Spoofed IP

“Valid” Sequence IDs

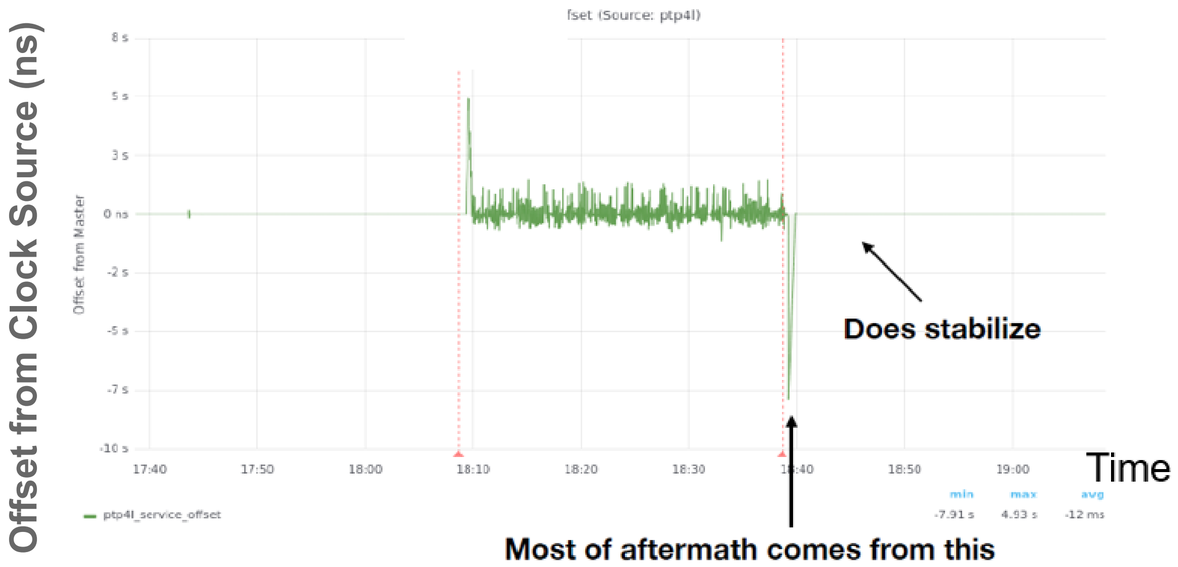
No need to spoof sequence IDs

200-300 spam packets/second

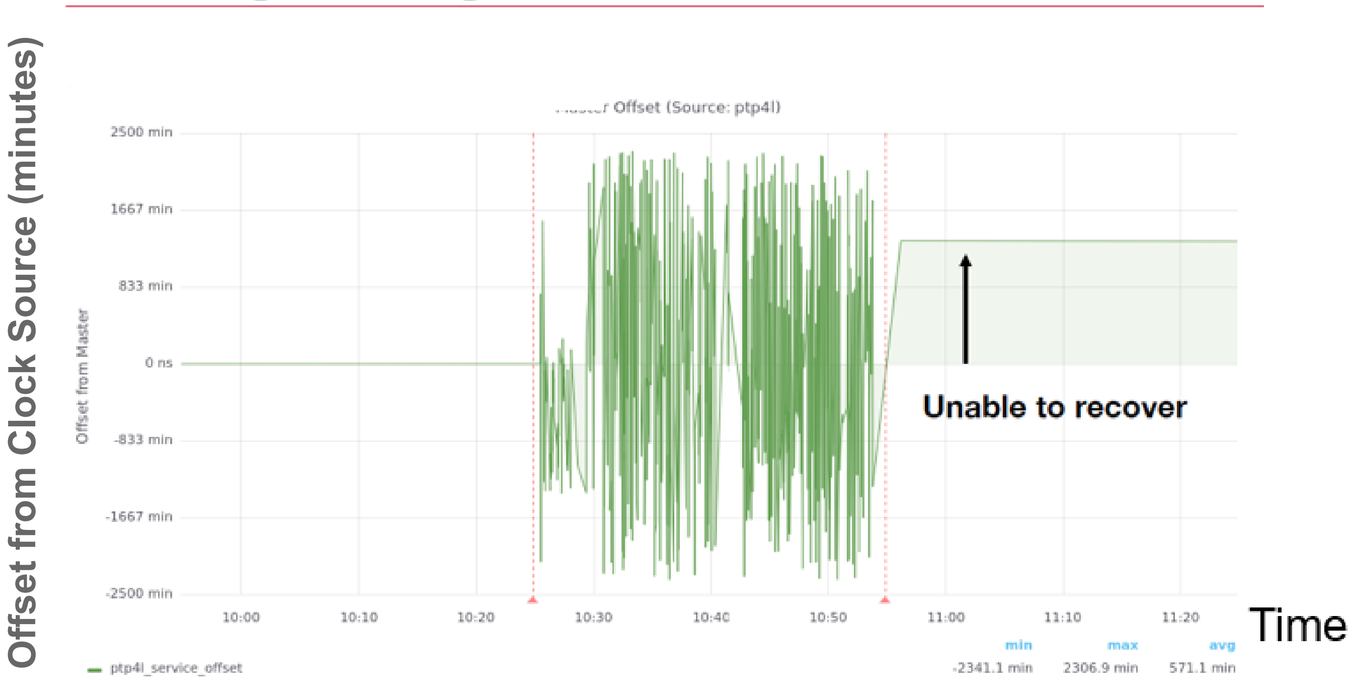
Average Offset During Attack: 137.8 ms

Average Offset After Attack: -86.1 ms

Announce DoS - Graph



Source Spoof – pretend to be the main clock source and send false data to the followers



30 minute attack can push the clock days or years out of sync

We do not need to know the IP address of the follower since multicast is supported; the multicast address (224.0.1.129) and port (320) always remain the same.

The clock ID of the follower is not required.

We only need to know the MAC address of the PTP enabled switch.

Although the follower recognizes that something is wrong (as reflected in the syslog and management console logs), it still accepts our spoofed SYNC packets.

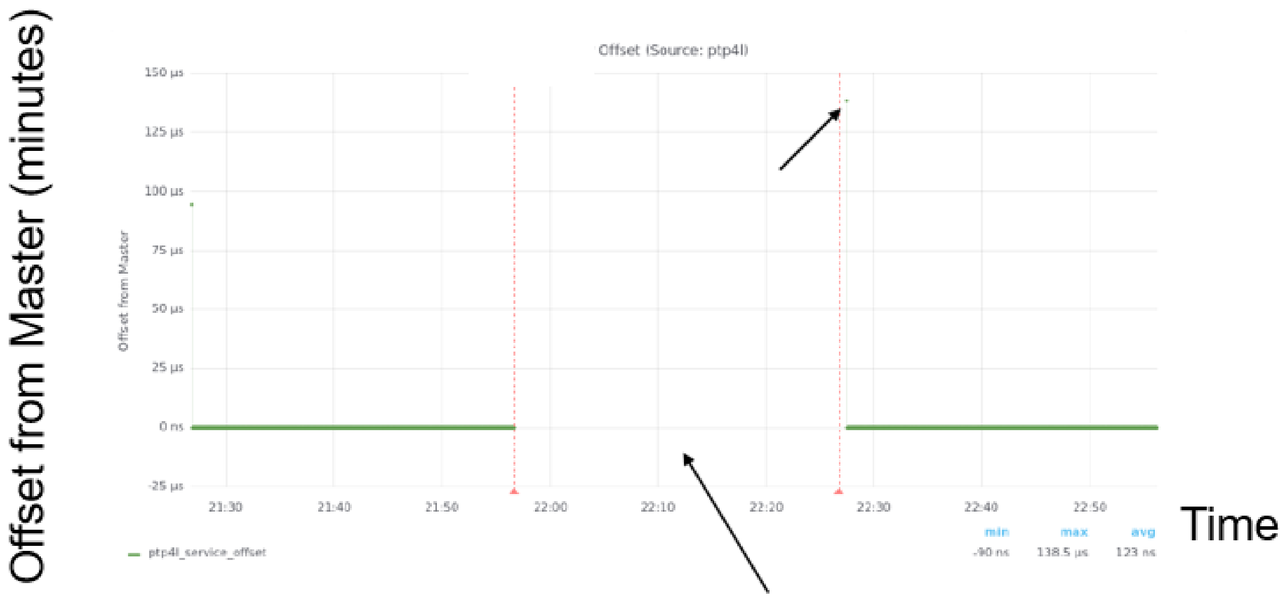
Atomic Source Takeover

Fake the whole PTP process and pretend to be an atomic clock

192.168.1.2	224.0.1.129	PTPv2	310	106	Announce Message
192.168.1.2	224.0.1.129	PTPv2	620	86	Sync Message
192.168.1.2	224.0.1.129	PTPv2	620	86	Follow_Up Message
192.168.1.3	224.0.1.129	PTPv2	2437	86	Delay_Req Message
192.168.1.2	224.0.1.129	PTPv2	2437	96	Delay_Resp Message
192.168.1.3	224.0.1.129	PTPv2	2438	86	Delay_Req Message
192.168.1.2	224.0.1.129	PTPv2	2438	96	Delay_Resp Message
192.168.1.2	224.0.1.129	PTPv2	621	86	Sync Message
192.168.1.2	224.0.1.129	PTPv2	621	86	Follow_Up Message
192.168.1.3	224.0.1.129	PTPv2	2439	86	Delay_Req Message
192.168.1.2	224.0.1.129	PTPv2	2439	96	Delay_Resp Message

Follower communicating with fake source

Full sync sequence



Average Offset During Attack: N/A

Average Offset After Attack: 148 ns

Acts like packets are being dropped

Clock Frequency Manipulation Attack

- Spoof packets with large amounts of data in correction field
- Clock frequency exceeds max value, unable to synchronize with source

```
ptp4l[96199.504]: master offset 814429228880942183 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.505]: master offset 814429228881957607 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.510]: master offset 814429228886727911 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.512]: master offset 814429228887743463 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.516]: master offset 814429228891974503 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.517]: master offset 814429228892993511 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.522]: master offset 814429228897199847 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.523]: master offset 814429228898218727 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.528]: master offset 814429228902661735 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.529]: master offset 814429228903680231 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.534]: master offset 814429228907889255 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.535]: master offset 814429228908909159 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.539]: master offset 814429228913116391 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.541]: master offset 814429228914133159 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.545]: master offset 814429228918348519 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.546]: master offset 814429228919375335 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.551]: master offset 814429228923794727 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.552]: master offset 814429228924813735 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.557]: master offset 814429228929037799 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.558]: master offset 814429228930058855 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.563]: master offset 814429228934266727 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.564]: master offset 814429228935297255 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.569]: master offset 814429228939722471 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.570]: master offset 814429228940744615 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.574]: master offset 814429228944960103 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.576]: master offset 814429228945980007 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.580]: master offset 814429228950188135 s2 freq    -nan path delay 814429218391406124
ptp4l[96199.581]: master offset 814429228951233895 s2 freq    -nan path delay 814429218391406124
```

```
ptp4l[96534.372]: clockcheck: clock jumped backward or running slower than expected!
ptp4l[96534.372]: master offset   -3171763 s0 freq    -nan path delay      6992
ptp4l[96534.576]: port 1: delay timeout
ptp4l[96534.576]: delay   filtered       6992   raw       5888
ptp4l[96535.396]: port 1: delay timeout
ptp4l[96535.397]: delay   filtered       6992   raw       9184
ptp4l[96535.473]: clockcheck: clock jumped backward or running slower than expected!
ptp4l[96535.473]: master offset   -3180595 s0 freq    -nan path delay      6992
ptp4l[96536.213]: port 1: delay timeout
ptp4l[96536.213]: delay   filtered       6992   raw       8032
ptp4l[96536.573]: clockcheck: clock jumped backward or running slower than expected!
ptp4l[96536.573]: master offset   -3189491 s0 freq    -nan path delay      6992
ptp4l[96537.673]: clockcheck: clock jumped backward or running slower than expected!
ptp4l[96537.673]: master offset   -3198323 s0 freq    -nan path delay      6992
ptp4l[96537.675]: port 1: delay timeout
ptp4l[96537.675]: delay   filtered       6992   raw       5056
ptp4l[96538.249]: port 1: delay timeout
ptp4l[96538.249]: delay   filtered       7360   raw       7392
ptp4l[96538.773]: clockcheck: clock jumped backward or running slower than expected!
ptp4l[96538.773]: master offset   -3207587 s0 freq    -nan path delay      7360
ptp4l[96539.207]: port 1: delay timeout
ptp4l[96539.208]: delay   filtered       7056   raw       6784
ptp4l[96539.873]: clockcheck: clock jumped backward or running slower than expected!
ptp4l[96539.873]: master offset   -3216051 s0 freq    -nan path delay      7056
ptp4l[96540.973]: clockcheck: clock jumped backward or running slower than expected!
ptp4l[96540.974]: master offset   -3224947 s0 freq    -nan path delay      7056
ptp4l[96541.125]: port 1: delay timeout
ptp4l[96541.125]: delay   filtered       7056   raw       5664
ptp4l[96541.497]: port 1: delay timeout
```

Conclusions

- Covert Communication Channels for PTP demonstrated experimentally
  - Correction Field as used in Delay_request field and
    Sync Followup field (non-colliding sequence IDs, i.e. enterprise profile)
  - Reserved Field
  - Detectable exfiltration in three other channels

- Three zero day vulnerabilities identified and attacks experimentally demonstrated
  - Intermittent temporal vortex (inject data GM clock ID / accuracy fields)
  - MITM injection attack on the correction field
    (injects packets to introduce large follower clock offsets in boundary configurations)
    Disables PTP4L outputs, further investigation required
  - Clock frequency attack, i.e. Master Spoof Variation DoS
    (directly affects the clock frequency, not just the offset)

CCAC
MARIST