EMBEDDED LINUX CONFERENCE

# Kernel Review, Hard Mode

Finding bugs where there are none.

Ulrich Hecht, Civil Infrastructure Platform Project



# About this speaker

- Linux professional since 1998
- Kernel developer since 2013
- With the CIP Project since 2021,
  maintaining the oldest kernels money can buy.
- Official unofficial AI stooge since 2026

## How a self-maintained SLTS kernel is made (abridged version)

1. A tool dredges new patches and tries to apply them to SLTS.
   - Whatever does not apply gets left out for now.
2. **Review patches that fail to apply cleanly.**
   - Most are not applicable at all.
   - We backport the important ones.
3. **Review patches that *don't* fail to apply.** Sometimes a patch applies even though it shouldn't...

## How a self-maintained SLTS kernel is made (abridged version)

1. A tool dredges new patches and tries to apply them to SLTS.
   - Whatever does not apply gets left out for now.
2. **Review patches that fail to apply cleanly.**
   - Most are not applicable at all.
   - We backport the important ones.
3. Review patches that *don't* fail to apply. Sometimes a patch applies even though it shouldn't...

## How a self-maintained SLTS kernel is made (abridged version)

1. A tool dredges new patches and tries to apply them to SLTS.
   - Whatever does not apply gets left out for now.
2. Review patches that fail to apply cleanly.
   - Most are not applicable at all.
   - We backport the important ones.
3. Review patches that *don't* fail to apply. Sometimes a patch applies even though it shouldn't...

# So what’s so "hard" here?

# So what’s so "hard" here?

# So what’s so "hard" here?

- Incoming patches are solid.

# So what’s so "hard" here?

- Incoming patches are solid.
- Well-reviewed, tested

# So what's so "hard" here?

- Incoming patches are solid.
- Well-reviewed, tested
- Linux Kernel Seal of Approval

```
Reported-by: Xiao Liang <shaw.leon@gmail.com>
Closes: https://lore.kernel.org/netdev/CABAhCOSzP1vaThGV35_VnsRCb=87_CPjPV[ ... ]
Fixes: 6c742e714d8c ("ipip: add x-netns support")
Cc: stable@vger.kernel.org
Signed-off-by: Maoyi Xie <maoyixie.tju@gmail.com>
Reviewed-by: Kuniyuki Iwashima <kuniyu@google.com>
Link: https://patch.msgid.link/20260612085941.3158249-3-maoyixie.tju@gmail.com
Signed-off-by: Jakub Kicinski <kuba@kernel.org>
Signed-off-by: Sasha Levin <sashal@kernel.org>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
```



# So what's so "hard" here?

- Incoming patches are solid.
- Well-reviewed, tested
- Linux Kernel Seal of Approval




```
Reported-by: Xiao Liang <shaw.leon@gmail.com>
Closes: https://lore.kernel.org/netdev/CABAhCOSzP1vaThGV35_VnsRCb=87_CPjPV[ ... ]
Fixes: 6c742e714d8c ("ipip: add x-netns support")
Cc: stable@vger.kernel.org
Signed-off-by: Maoyi Xie <maoyixie.tju@gmail.com>
Reviewed-by: Kuniyuki Iwashima <kuniyu@google.com>
Link: https://patch.msgid.link/20260612085941.3158249-3-maoyixie.tju@gmail.com
Signed-off-by: Jakub Kicinski <kuba@kernel.org>
Signed-off-by: Sasha Levin <sashal@kernel.org>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
```

^ for mainline...

# Why it's hard: Volume!



# Why it’s hard: Volume!

- There are a lot of patches incoming



# Why it's hard: Volume!

- There are a lot of patches incoming
- There have always been a lot of patches incoming



# Why it's hard: Volume!

- There are a lot of patches incoming
- There have always been a lot of patches incoming
- There are even more patches incoming now



# Why it's hard: Volume!

- There are a lot of patches incoming
- There have always been a lot of patches incoming
- There are even more patches incoming now




There were 1300 patches incoming for the last 4.19-cip release.

# Why it's hard: Volume!

- There are a lot of patches incoming
- There have always been a lot of patches incoming
- There are even more patches incoming now




There were 1300 patches incoming for the last 4.19-cip release.

There are already (as of writing) 900 patches queued for the next release.

# Why it’s hard: Quality.



# Why it’s hard: Quality.

- Most patches look perfect. No beginner’s mistakes.



# Why it’s hard: Quality.

- Most patches look perfect. No beginner’s mistakes.
- No thousands of lines of code with tons of trivial issues.



# Why it’s hard: Quality.

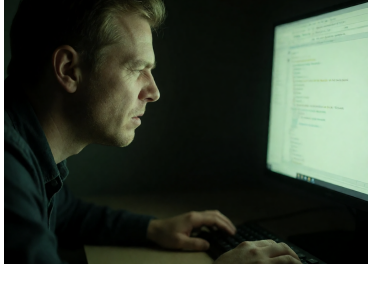
- Most patches look perfect. No beginner’s mistakes.
- No thousands of lines of code with tons of trivial issues.
- Compact. Concise. High context.



# Why it’s hard: Quality.

- Most patches look perfect. No beginner’s mistakes.
- No thousands of lines of code with tons of trivial issues.
- Compact. Concise. High context.



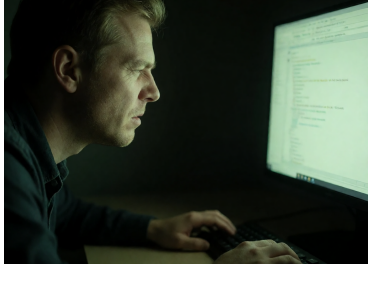



# Why it's hard: Quality.

- Most patches look perfect. No beginner's mistakes.
- No thousands of lines of code with tons of trivial issues.
- Compact. Concise. High context.

You have reviewed hundreds of patches like these...



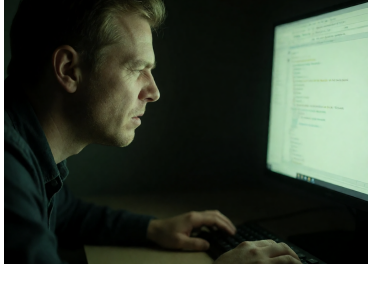
# Why it's hard: Quality.

- Most patches look perfect. No beginner's mistakes.
- No thousands of lines of code with tons of trivial issues.
- Compact. Concise. High context.

You have reviewed hundreds of patches like these...

...and then there's the one patch that is BAD!



# Why it's hard: Quality.

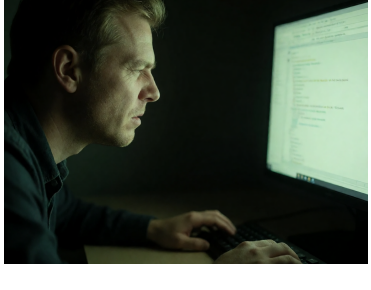
- Most patches look perfect. No beginner's mistakes.
- No thousands of lines of code with tons of trivial issues.
- Compact. Concise. High context.




You have reviewed hundreds of patches like these...

...and then there's the one patch that is BAD!

Will you spot it?



# "Perfect" Example

```
Author: Wesley Cheng <quic_wcheng@quicinc.com>
Date:   Thu Sep 1 12:36:23 2022 -0700

    usb: dwc3: Increase DWC3 controller halt timeout

    [ Upstream commit 461ee467507cb98a348fa91ff8460908bb0ea423 ]

    Since EP0 transactions need to be completed before the
    controller halt sequence is finished, this may take some time
    depending on the host and the enabled functions. Increase the
    controller halt timeout, so that we give the controller
    sufficient time to handle EP0 transfers.

    Signed-off-by: Wesley Cheng <quic_wcheng@quicinc.com>
    Link: [ ... ]
    Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
    Stable-dep-of: d3a8c28426fc ("usb: dwc3: Fix timeout issue during
      controller enter/exit from halt state")
    Signed-off-by: Sasha Levin <sashal@kernel.org>
```

```
diff --git a/drivers/usb/dwc3/gadget.c b/drivers/usb/dwc3/gadget.c
index 9c2b9e0886cb9..fe210c7eeee8c 100644
--- a/drivers/usb/dwc3/gadget.c
+++ b/drivers/usb/dwc3/gadget.c
@@ -1763,7 +1763,7 @@ static int dwc3_gadget_set_selfpowered(struct usb_gadget *g,
 static int dwc3_gadget_run_stop(struct dwc3 *dwc, int is_on, int suspend)
 {
        u32                     reg;
-       u32                     timeout = 500;
+       u32                     timeout = 2000;
        u32                     saved_config = 0;

        if (pm_runtime_suspended(dwc->dev))
@@ -1823,6 +1823,7 @@ static int dwc3_gadget_run_stop(struct dwc3 *dwc, int is_on,
                                                        int suspend)
        dwc3_writel(dwc->regs, DWC3_DCTL, reg);

        do {
+               usleep_range(1000, 2000);
                reg = dwc3_readl(dwc->regs, DWC3_DSTS);
                reg &= DWC3_DSTS_DEVCTRLHLT;
        } while (--timeout && !(!is_on ^ !reg));
```

# "Perfect" Example

```
Author: Wesley Cheng <quic_wcheng@quicinc.com>
Date:   Thu Sep 1 12:36:23 2022 -0700

    usb: dwc3: Increase DWC3 controller halt timeout

    [ Upstream commit 461ee467507cb98a348fa91ff8460908bb0ea423 ]

    Since EP0 transactions need to be completed before the
    controller halt sequence is finished, this may take some time
    depending on the host and the enabled functions. Increase the
    controller halt timeout, so that we give the controller
    sufficient time to handle EP0 transfers.

    Signed-off-by: Wesley Cheng <quic_wcheng@quicinc.com>
    Link: [ ... ]
    Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
    Stable-dep-of: d3a8c28426fc ("usb: dwc3: Fix timeout issue during
      controller enter/exit from halt state")
    Signed-off-by: Sasha Levin <sashal@kernel.org>
```

```
diff --git a/drivers/usb/dwc3/gadget.c b/drivers/usb/dwc3/gadget.c
index 9c2b9e0886cb9..fe210c7eeee8c 100644
--- a/drivers/usb/dwc3/gadget.c
+++ b/drivers/usb/dwc3/gadget.c
@@ -1763,7 +1763,7 @@ static int dwc3_gadget_set_selfpowered(struct usb_gadget *g,
 static int dwc3_gadget_run_stop(struct dwc3 *dwc, int is_on, int suspend)
 {
        u32                     reg;
-       u32                     timeout = 500;
+       u32                     timeout = 2000;
        u32                     saved_config = 0;

        if (pm_runtime_suspended(dwc->dev))
@@ -1823,6 +1823,7 @@ static int dwc3_gadget_run_stop(struct dwc3 *dwc, int is_on,
                                                        int suspend)
        dwc3_writel(dwc->regs, DWC3_DCTL, reg);

        do {
+               usleep_range(1000, 2000);
                reg = dwc3_readl(dwc->regs, DWC3_DSTS);
                reg &= DWC3_DSTS_DEVCTRLHLT;
        } while (--timeout && !(!is_on ^ !reg));
```

In *mainline* this is fine...

# "Perfect" Example

```
Author: Wesley Cheng <quic_wcheng@quicinc.com>
Date:   Thu Sep 1 12:36:23 2022 -0700

    usb: dwc3: Increase DWC3 controller halt timeout

    [ Upstream commit 461ee467507cb98a348fa91ff8460908bb0ea423 ]

    Since EP0 transactions need to be completed before the
    controller halt sequence is finished, this may take some time
    depending on the host and the enabled functions. Increase the
    controller halt timeout, so that we give the controller
    sufficient time to handle EP0 transfers.

    Signed-off-by: Wesley Cheng <quic_wcheng@quicinc.com>
    Link: [ ... ]
    Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>
    Stable-dep-of: d3a8c28426fc ("usb: dwc3: Fix timeout issue during
      controller enter/exit from halt state")
    Signed-off-by: Sasha Levin <sashal@kernel.org>
```

```
diff --git a/drivers/usb/dwc3/gadget.c b/drivers/usb/dwc3/gadget.c
index 9c2b9e0886cb9..fe210c7eeee8c 100644
--- a/drivers/usb/dwc3/gadget.c
+++ b/drivers/usb/dwc3/gadget.c
@@ -1763,7 +1763,7 @@ static int dwc3_gadget_set_selfpowered(struct usb_gadget *g,
 static int dwc3_gadget_run_stop(struct dwc3 *dwc, int is_on, int suspend)
 {
        u32                     reg;
-       u32                     timeout = 500;
+       u32                     timeout = 2000;
        u32                     saved_config = 0;

        if (pm_runtime_suspended(dwc->dev))
@@ -1823,6 +1823,7 @@ static int dwc3_gadget_run_stop(struct dwc3 *dwc, int is_on,
                                                        int suspend)
        dwc3_writel(dwc->regs, DWC3_DCTL, reg);

        do {
+               usleep_range(1000, 2000);
                reg = dwc3_readl(dwc->regs, DWC3_DSTS);
                reg &= DWC3_DSTS_DEVCTRLHLT;
        } while (--timeout && !(!is_on ^ !reg));
```

In *mainline* this is fine... In 4.19-cip it's LETHAL!

# The Revolution can be downloaded?

# The Revolution can be downloaded?

# The Revolution can be downloaded?

- It’s the good old year 2025...

# The Revolution can be downloaded?

- It's the good old year 2025...
- There's a new *"AI"* thing, "Qwen with Questions"

# The Revolution can be downloaded?

- It's the good old year 2025...
- There's a new *"AI"* thing, "Qwen with Questions"
- "Reasoning"?



# The Revolution can be downloaded?

- It's the good old year 2025...
- There's a new *"AI"* thing, "Qwen with Questions"
- "Reasoning"?
- May be useful it to tell apart scary and harmless patches, so we don't have to look at everything. Let's ask it:



# The Revolution can be downloaded?

- It's the good old year 2025...
- There's a new *"AI"* thing, "Qwen with Questions"
- "Reasoning"?
- May be useful it to tell apart scary and harmless patches, so we don't have to look at everything. Let's ask it:

> Check if this patch makes changes that may cause regressions.



# The Revolution can be downloaded!

- What QwQ-32B said: (verbatim)

# The Revolution can be downloaded!

- What QwQ-32B said: (verbatim)

# The Revolution can be downloaded!

- What QwQ-32B said: (verbatim)

> The patch introduces incorrect manual decrements of `sch->q.qlen` in `netem_dequeue`, leading to potential undercounts and regressions in backlog tracking.



# The Revolution can be downloaded!

- What QwQ-32B said: (verbatim)

> The patch introduces incorrect manual decrements of `sch->q.qlen` in `netem_dequeue`, leading to potential undercounts and regressions in backlog tracking.

That isn't even what I asked about, that's...



# The Revolution can be downloaded!

- What QwQ-32B said: (verbatim)

> The patch introduces incorrect manual decrements of `sch->q.qlen` in `netem_dequeue`, leading to potential undercounts and regressions in backlog tracking.

That isn't even what I asked about, that's...



- That's from my own backport!

# The Revolution can be downloaded!

- What QwQ-32B said: (verbatim)

> The patch introduces incorrect manual decrements of `sch->q.qlen` in `netem_dequeue`, leading to potential undercounts and regressions in backlog tracking.



That isn't even what I asked about, that's...

- That's from my own backport!
- That's not just a vague guess!

# The Revolution can be downloaded!

- What QwQ-32B said: (verbatim)

> The patch introduces incorrect manual decrements of `sch->q.qlen` in `netem_dequeue`, leading to potential undercounts and regressions in backlog tracking.



That isn't even what I asked about, that's...

- That's from my own backport!
- That's not just a vague guess!
- That's actually, factually, indisputably *correct!*

# Harnessing the power of the ~~atom~~ LLM: SLTR

# Harnessing the power of the ~~atom~~ LLM: SLTR

# Harnessing the power of the ~~atom~~ LLM: SLTR

- "Super Long Term Review"



# Harnessing the power of the ~~atom~~ LLM: SLTR

- "Super Long Term Review"
- 1200 lines of Python (all written by hand :))



# Harnessing the power of the ~~atom~~ LLM: SLTR

- "Super Long Term Review"
- 1200 lines of Python (all written by hand :))
- ~800 lines of prompts (~250 per model)



# Harnessing the power of the ~~atom~~ LLM: SLTR

- "Super Long Term Review"
- 1200 lines of Python (all written by hand :))
- ~800 lines of prompts (~250 per model)
- Feeds a given range of kernel commits to an LLM:

> "Here's a patch, check what's broken." (very abridged)



# Harnessing the power of the ~~atom~~ LLM: SLTR

- "Super Long Term Review"
- 1200 lines of Python (all written by hand :))
- ~800 lines of prompts (~250 per model)
- Feeds a given range of kernel commits to an LLM:

  > "Here's a patch, check what's broken." (very abridged)

- Aggregates the results into an HTML report



# Harnessing the power of the ~~atom~~ LLM: SLTR

- "Super Long Term Review"
- 1200 lines of Python (all written by hand :))
- ~800 lines of prompts (~250 per model)
- Feeds a given range of kernel commits to an LLM:

  > "Here's a patch, check what's broken." (very abridged)

- Aggregates the results into an HTML report
- Most important: Keeps the LLM on a short leash



# SLTR Report

| 200 lines | 200 lines | 200 lines | | |
|---|---|---|---|---|
| APPLY[T] | APPLY[T] | APPLY[T] | b1ec593e6ca5f | net: ipip: require CAP_NET_ADMIN in the device netns for changelink |
| APPLY[T] | APPLY[T] | APPLY[T] | 39fbc8d0433ac | net: ip6_vti: require CAP_NET_ADMIN in the device netns for changelink |
| APPLY[T] | APPLY[T] | APPLY[T] | 661ea88907dbd | net: sit: require CAP_NET_ADMIN in the device netns for changelink |
| APPLY[T] | APPLY[T] | APPLY[T] | 17e02efe92daa | net: ip6_tunnel: require CAP_NET_ADMIN in the device netns for changelink |
| APPLY[T] | APPLY[T] | APPLY[T] | 8ef52a2fd7e11 | net: ip_vti: require CAP_NET_ADMIN in the device netns for changelink |
| APPLY[T] | APPLY[T] | APPLY[T] | 85c1fd0e74525 | NFS: Pin the 'struct nfs_server' during a FREE_STATEID call |
| REJECT[T] | REJECT[T] | REJECT[T] | 2ac51ae0055fb | wifi: carl9170: fix buffer overflow in rx_stream failover path |
| FAIL[T] | APPLY[T] | APPLY[T] | 4c07227affb83 | ocfs2: reject non-inline dinodes with i_size and zero i_clusters |
| APPLY[T] | APPLY[T] | APPLY[T] | 26630a797613b | nilfs2: fix incorrect nilfs_error() call |
| APPLY[T] | APPLY[T] | APPLY[T] | c7b2d70045dd2 | wifi: libertas_tf: fix use-after-free in lbtf_free_adapter() |
| APPLY[T] | APPLY[T] | APPLY[T] | 9a2bc010a110a | net: remove WARN_ON_ONCE() from sk_mc_loop() |

# SLTR Report

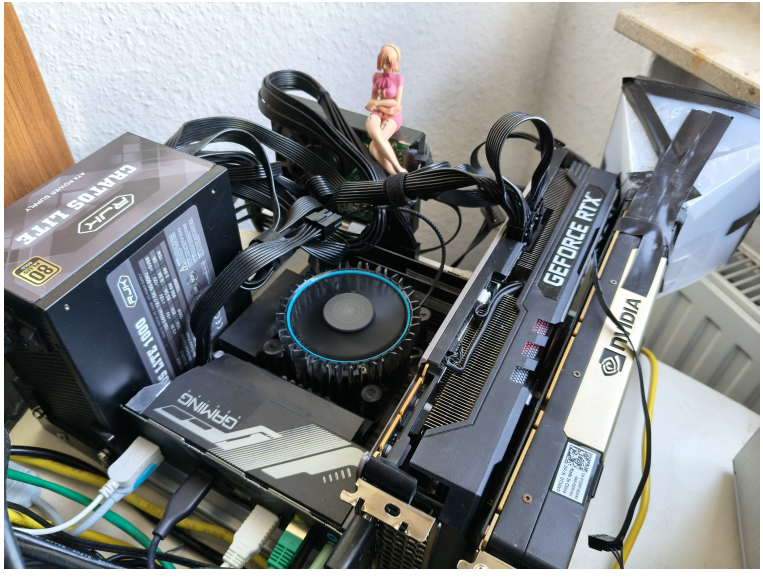
| 200 lines | 200 lines | 200 lines | | |
|---|---|---|---|---|
| APPLY[T] | APPLY[T] | APPLY[T] | b1ec593e6ca5f | net: ipip: require CAP_NET_ADMIN in the device netns for changelink |
| APPLY[T] | APPLY[T] | APPLY[T] | 39fbc8d0433ac | net: ip6_vti: require CAP_NET_ADMIN in the device netns for changelink |
| APPLY[T] | APPLY[T] | APPLY[T] | 661ea88907dbd | net: sit: require CAP_NET_ADMIN in the device netns for changelink |
| APPLY[T] | APPLY[T] | APPLY[T] | 17e02efe92daa | net: ip6_tunnel: require CAP_NET_ADMIN in the device netns for changelink |
| APPLY[T] | APPLY[T] | APPLY[T] | 8ef52a2fd7e11 | net: ip_vti: require CAP_NET_ADMIN in the device netns for changelink |
| APPLY[T] | APPLY[T] | APPLY[T] | 85c1fd0e74525 | NFS: Pin the 'struct nfs_server' during a FREE_STATEID call |
| REJECT[T] | REJECT[T] | REJECT[T] | 2ac51ae0055fb | wifi: carl9170: fix buffer overflow in rx_stream failover path |

I verified the key facts with the tools: `rx_failover_missing` is a signed `int` (carl9170.h:421), it is assigned as `clen - tlen` where both `clen` and `tlen` are `unsigned int` (so it frequently wraps to a negative value), the failover SKB is exactly `AR9170_RX_STREAM_MAX_SIZE` = 0xffff bytes, each URB is at most 32768 bytes, and `min_t(type,x,y)` casts both operands to `type` (include/linux/kernel.h:765).

Analysis of the only changed statement `memcpy(skb_put(ar->rx_failover, min_t(unsigned int, tlen, ar->rx_failover_missing)), ...)`:
- When `rx_failover_missing` is negative (a normal value, e.g. clen in [32765,32767] with a full 32768-byte first transfer gives `clen - tlen` = -3), the `unsigned int` cast turns it into a huge value, so `min_t` returns the full `tlen`. The copy is NOT capped, `tlen_1 + tlen_2` can still reach 65536 > 65535, and `skb_over_panic` still fires — the reported bug is not actually fixed in that case.
- When `rx_failover_missing` is positive, the cap works and the SKB stays ≤ clen, but the code then copies only the continuation bytes and immediately `return`s without ever processing `tbuf + copied, tlen - copied`. The original appended the *entire* second transfer to the SKB and processed it, so any complete frames following the split frame in that URB were handled; the patch silently drops them — a data-loss regression.
- The unchanged `ar->rx_failover_missing -= tlen;` also now subtracts the wrong amount (tlen instead of the capped copy), only masked by the later reset-to-zero.

The change therefore neither robustly fixes the overflow nor preserves the previous frame-processing behavior.

**Action:** REJECT




# Hardware

- RTX 3090 24GB
- V100 PCIe 32GB
- 64GB DDR4 RAM
- 1 kW PSU
- a CPU (x64, probably)
- electrical tape
- plastic folder lid
- high-pressure case fan

# Why use local LLMs?

# Why use local LLMs?

- Stability

# Why use local LLMs?

- Stability
- Tamper-proofing

# Why use local LLMs?

- Stability
- Tamper-proofing
- Vendor independence

# Why use local LLMs?

- Stability
- Tamper-proofing
- Vendor independence
- Technical requirements:

# Why use local LLMs?

- Stability
- Tamper-proofing
- Vendor independence
- Technical requirements:
  - Introspection

# Why use local LLMs?

- Stability
- Tamper-proofing
- Vendor independence
- Technical requirements:
  - Introspection
  - Complete context control

# Why use local LLMs?

- Stability
- Tamper-proofing
- Vendor independence
- Technical requirements:
  - Introspection
  - Complete context control
- ...and the other 10000 (geopolitical, economical, price variability, availability...

# How SLTR tames the LLM: Containment

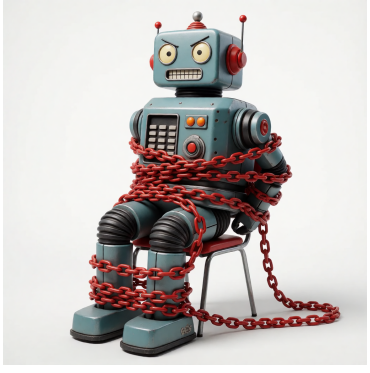
# How SLTR tames the LLM: Containment

Remember the OpenAI/HuggingFace hack?

# How SLTR tames the LLM: Containment

Remember the OpenAI/HuggingFace hack?

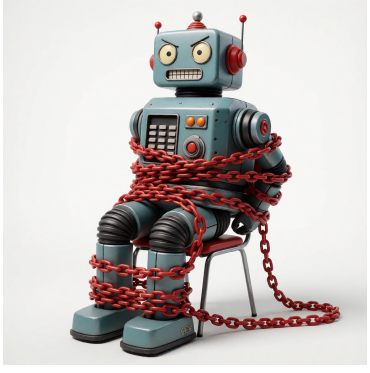
AI agents need to be kept on a very short leash!



# How SLTR tames the LLM: Containment

Remember the OpenAI/HuggingFace hack?

AI agents need to be kept on a very short leash!

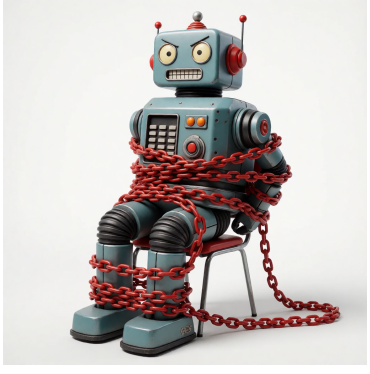


# How SLTR tames the LLM: Containment

Remember the OpenAI/HuggingFace hack?

AI agents need to be kept on a very short leash!

- **No write access** to anything.

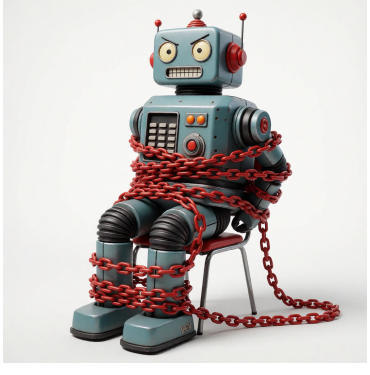


# How SLTR tames the LLM: Containment

Remember the OpenAI/HuggingFace hack?

AI agents need to be kept on a very short leash!

- **No write access** to anything.
- Limited read access:

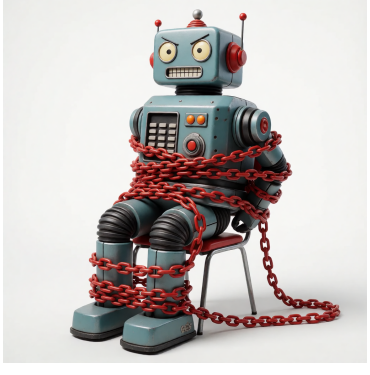


# How SLTR tames the LLM: Containment

Remember the OpenAI/HuggingFace hack?

AI agents need to be kept on a very short leash!

- **No write access** to anything.
- Limited read access:
  - Specific git repo, specific branches.



# How SLTR tames the LLM: Containment



Remember the OpenAI/HuggingFace hack?

AI agents need to be kept on a very short leash!

- **No write access** to anything.
- Limited read access:
  - Specific git repo, specific branches.
  - Keeps the LLM honest[citation needed] and focused.

# How SLTR tames the LLM: Control the Narrative

# How SLTR tames the LLM: Control the Narrative

- LLMs don’t know what’s important.

# How SLTR tames the LLM: Control the Narrative

- LLMs don’t know what’s important.
- LLMs will use *any* excuse to take a shortcut!

# How SLTR tames the LLM: Control the Narrative

- LLMs don't know what's important.
- LLMs will use *any* excuse to take a shortcut!
- *Open-weight* LLMs are not pathological sycophants, but...

# How SLTR tames the LLM: Control the Narrative

- LLMs don’t know what’s important.
- LLMs will use *any* excuse to take a shortcut!
- *Open-weight* LLMs are not pathological sycophants, but…

1. Scrub anything that indicates this patch is "good".

# How SLTR tames the LLM: Control the Narrative

- LLMs don’t know what’s important.
- LLMs will use *any* excuse to take a shortcut!
- *Open-weight* LLMs are not pathological sycophants, but...

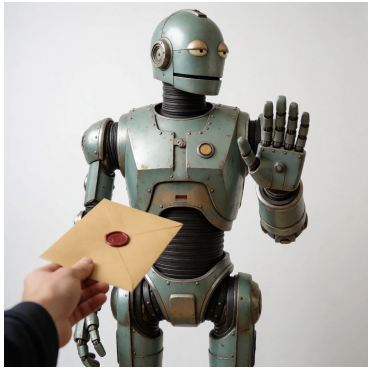
1. Scrub anything that indicates this patch is "good".
2. Scrub *everything* that has anything to do with people!

# How SLTR tames the LLM: Make it do what you say

# How SLTR tames the LLM: Make it do what you say

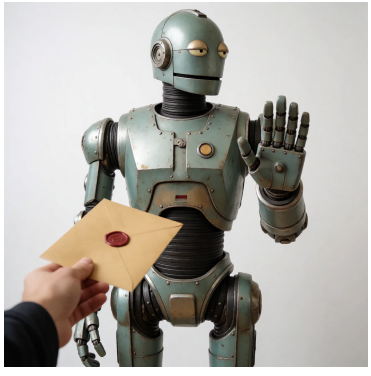
# How SLTR tames the LLM: Make it do what you say

- LLMs vary when it comes to prompt following.



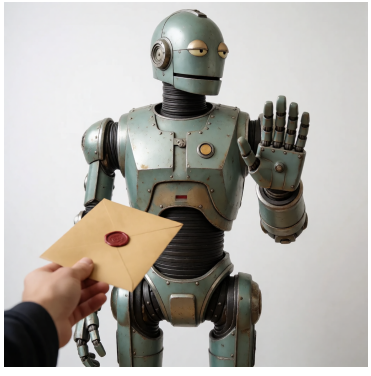
# How SLTR tames the LLM: Make it do what you say

- LLMs vary when it comes to prompt following.
- Mitigation: Use Legalese.



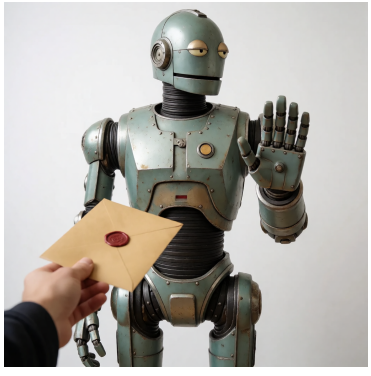
# How SLTR tames the LLM: Make it do what you say

- LLMs vary when it comes to prompt following.
- Mitigation: Use Legalese.
  - "shall", "must", "must not". Avoid pronouns.



# How SLTR tames the LLM: Make it do what you say

- LLMs vary when it comes to prompt following.
- Mitigation: Use Legalese.
  - "shall", "must", "must not". Avoid pronouns.
- Sometimes the best way forward is not to care.



# How SLTR tames the LLM: Tools

# How SLTR tames the LLM: Tools

# How SLTR tames the LLM: Tools

- The olden days (last year): We rely on guesswork

# How SLTR tames the LLM: Tools

- The olden days (last year): We rely on guesswork
- The modern age: Tool calls



# How SLTR tames the LLM: Tools

- The olden days (last year): We rely on guesswork
- The modern age: Tool calls
  - LLMs inspect the actual code as needed.



# How SLTR tames the LLM: Tools

- The olden days (last year): We rely on guesswork
- The modern age: Tool calls
  - LLMs inspect the actual code as needed.
  - Eliminates guesswork, provides solid truths.



# How SLTR tames the LLM: Tools

- The olden days (last year): We rely on guesswork
- The modern age: Tool calls
  - LLMs inspect the actual code as needed.
  - Eliminates guesswork, provides solid truths.
- Complex tools?



# How SLTR tames the LLM: Tools

- The olden days (last year): We rely on guesswork
- The modern age: Tool calls
  - LLMs inspect the actual code as needed.
  - Eliminates guesswork, provides solid truths.
- Complex tools?
  - difficult to keep up-to-date (and often just broken)



# How SLTR tames the LLM: Tools

- The olden days (last year): We rely on guesswork
- The modern age: Tool calls
  - LLMs inspect the actual code as needed.
  - Eliminates guesswork, provides solid truths.
- Complex tools?
  - difficult to keep up-to-date (and often just broken)
- Simple tools!



# How SLTR tames the LLM: Tools

- The olden days (last year): We rely on guesswork
- The modern age: Tool calls
  - LLMs inspect the actual code as needed.
  - Eliminates guesswork, provides solid truths.
- Complex tools?
  - difficult to keep up-to-date (and often just broken)
- Simple tools!
  - use git; reliable, no maintenance

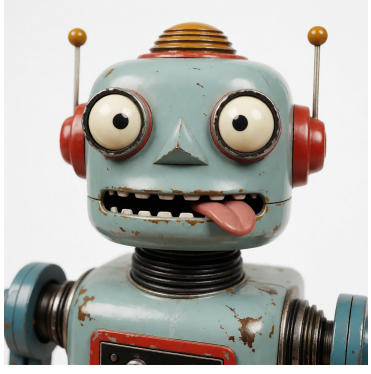


# How SLTR tames the LLM: Pathologies

# How SLTR tames the LLM: Pathologies

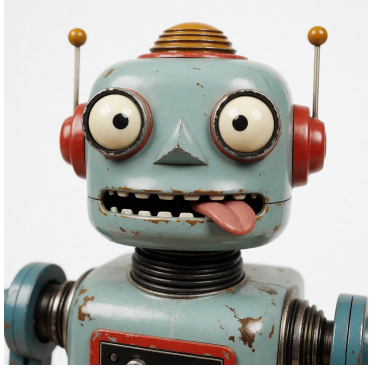
# How SLTR tames the LLM: Pathologies

- Most (all?) LLMs have some sort of pathology:



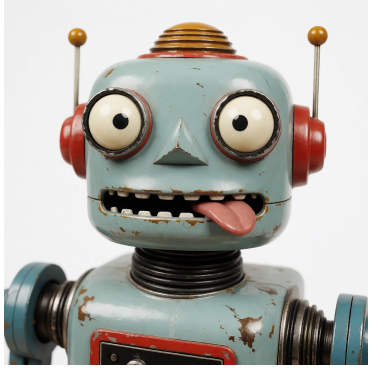
# How SLTR tames the LLM: Pathologies

- Most (all?) LLMs have some sort of pathology:
  - "hallucinations" (aka making shit up)



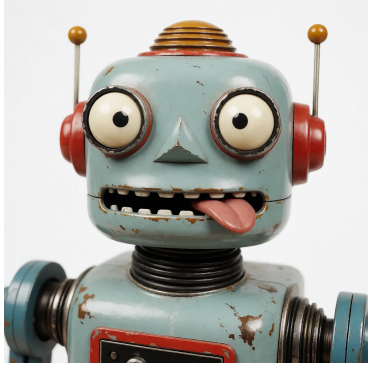
# How SLTR tames the LLM: Pathologies

- Most (all?) LLMs have some sort of pathology:
  - "hallucinations" (aka making shit up)
  - not able to handle their own syntax



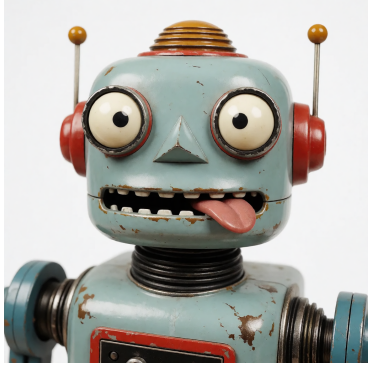
# How SLTR tames the LLM: Pathologies

- Most (all?) LLMs have some sort of pathology:
  - "hallucinations" (aka making shit up)
  - not able to handle their own syntax
  - tool call phobia



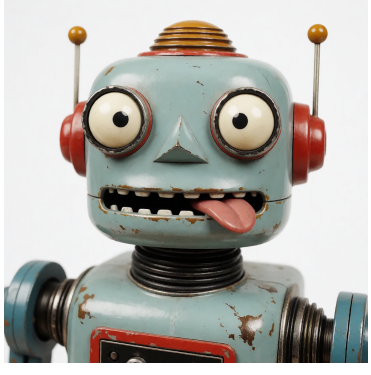
# How SLTR tames the LLM: Pathologies

- Most (all?) LLMs have some sort of pathology:
  - "hallucinations" (aka making shit up)
  - not able to handle their own syntax
  - tool call phobia
  - endless loops



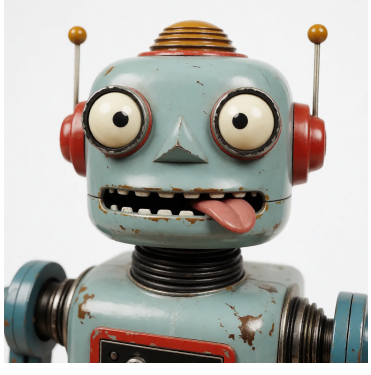
# How SLTR tames the LLM: Pathologies

- Most (all?) LLMs have some sort of pathology:
  - "hallucinations" (aka making shit up)
  - not able to handle their own syntax
  - tool call phobia
  - endless loops
  - quitting early



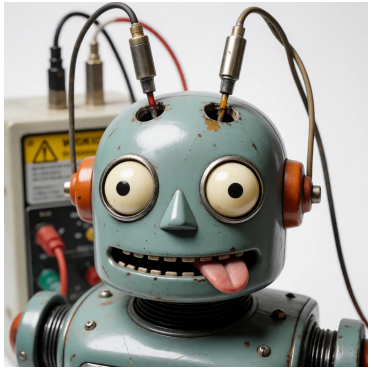
# How SLTR tames the LLM: Pathologies

- Most (all?) LLMs have some sort of pathology:
  - "hallucinations" (aka making shit up)
  - not able to handle their own syntax
  - tool call phobia
  - endless loops
  - quitting early



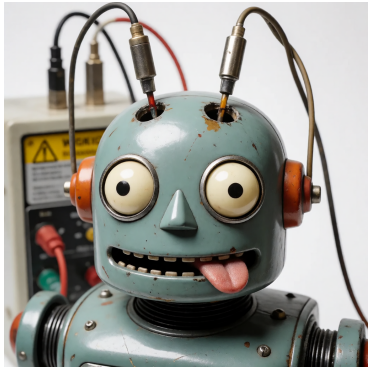
Notice how these are very regular?

# How SLTR tames the LLM: Needle in the Brain



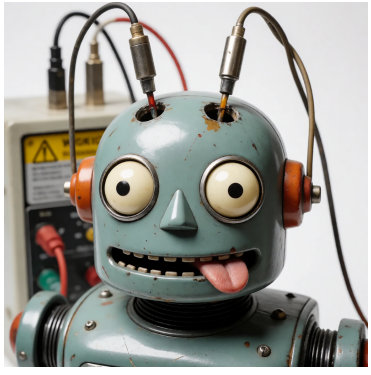
# How SLTR tames the LLM: Needle in the Brain

- We have control of the context.



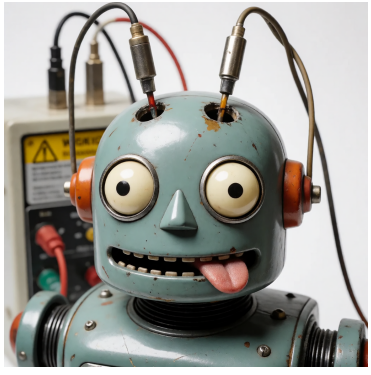
# How SLTR tames the LLM: Needle in the Brain

- We have control of the context.
- *Complete* control of the context.



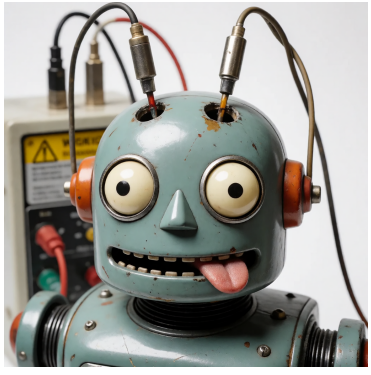
# How SLTR tames the LLM: Needle in the Brain

- We have control of the context.
- *Complete* control of the context.
- We know what the model "thinks".



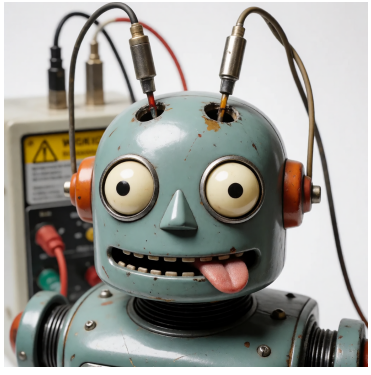
# How SLTR tames the LLM: Needle in the Brain

- We have control of the context.
- *Complete* control of the context.
- We know what the model "thinks".
- We can change what the model "thinks".



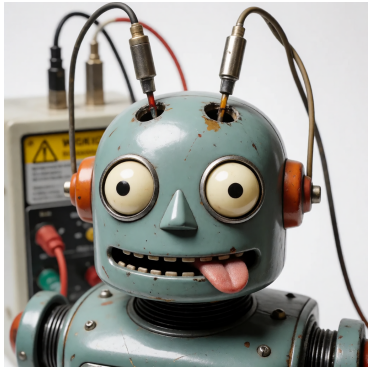
# How SLTR tames the LLM: Needle in the Brain

- We have control of the context.
- *Complete* control of the context.
- We know what the model "thinks".
- We can change what the model "thinks".
  - Model: "Uh, let me just make something up!"



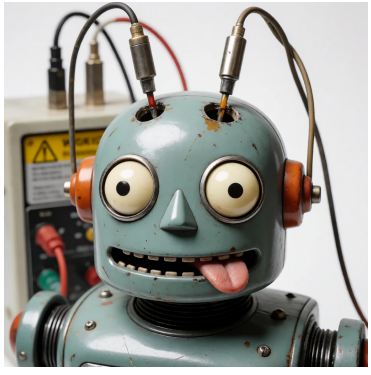
# How SLTR tames the LLM: Needle in the Brain

- We have control of the context.
- *Complete* control of the context.
- We know what the model "thinks".
- We can change what the model "thinks".
  - Model: "Uh, let me just make something up!"
  - SLTR: *BZZZZZT!*

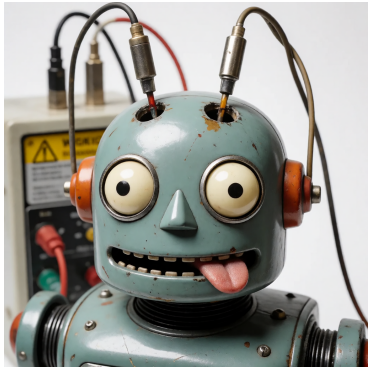


# How SLTR tames the LLM: Needle in the Brain

- We have control of the context.
- *Complete* control of the context.
- We know what the model "thinks".
- We can change what the model "thinks".
  - Model: "Uh, let me just make something up!"
  - SLTR: *BZZZZZT!*
  - Model: "~~Uh, let me just make something up!~~<tool_call>"



# How SLTR tames the LLM: Needle in the Brain



- We have control of the context.
- *Complete* control of the context.
- We know what the model "thinks".
- We can change what the model "thinks".
  - Model: "Uh, let me just make something up!"
  - SLTR: *BZZZZZT!*
  - Model: "~~Uh, let me just make something up!~~<tool_call>"

Very fast. Very reliable. Open-weight LLM **required**!

# How SLTR tames the LLM: Corrections Example

```
{
    '\n</think>\n': [
        "OK, I think I need to stop going in circles and just make a decision.",
        "Final answer.",
    ],
    '\n</think>\n\n<tool_call>\n': [
        "Actually, I just remembered",
        "I vaguely remember",
        "Actually, I recall",
        "Actually, I remember!",
        "Actually, I remember:",
        "Let me try to recall",
        "Let me recall",
        "Let's recall",
        "Actually, let me recall",
        "Hmm, I recall",
        "let me recall",
        "I remember now:",
        "Now I remember",
        "Now I recall",
        "Let me search memory:",
        "let me try to actually remember",
        "I'm starting to recall",
        "I just remembered!",
        "actual upstream history",
        "actually, I recall",
    ],
}
```

# Does it work?

# Does it work?

# Does it work?

- Shockingly well.



# Does it work?

- Shockingly well.
- Test set: 22 known-good and 16 known-bad patches



# Does it work?

- Shockingly well.
- Test set: 22 known-good and 16 known-bad patches
- Qwen 3.8 has no false positives.



# Does it work?

- Shockingly well.
- Test set: 22 known-good and 16 known-bad patches
- Qwen 3.8 has no false positives.
- Qwen 3.8 has no (hard) false negatives

  (What's a patch that claims to fix a bug, actually implements a NOP, all while the bug doesn't actually exist?)



# Does it work?

- Shockingly well.
- Test set: 22 known-good and 16 known-bad patches
- Qwen 3.8 has no false positives.
- Qwen 3.8 has no (hard) false negatives

  (What's a patch that claims to fix a bug, actually implements a NOP, all while the bug doesn't actually exist?)



- In practice: Very few false positives. (Important!)

  People really, *REALLY* don't like it when robots pester them with their unfounded opinions...

# What does it cost?

# What does it cost?

# What does it cost?

- Test rig: Hardware ~2400 EUR, ~750W electrical

  (pre memory-crisis money)



# What does it cost?

- Test rig: Hardware ~2400 EUR, ~750W electrical

  (pre memory-crisis money)
- AWS bill: 196 USD (September)



# What does it cost?

- Test rig: Hardware ~2400 EUR, ~750W electrical (pre memory-crisis money)
- AWS bill: 196 USD (September)
- 4.19-cip136 took 8 GPU-days to review



# What does it cost?

- Test rig: Hardware ~2400 EUR, ~750W electrical (pre memory-crisis money)
- AWS bill: 196 USD (September)
- 4.19-cip136 took 8 GPU-days to review
- Very much depends on the workload and model



# What does it cost?

- Test rig: Hardware ~2400 EUR, ~750W electrical (pre memory-crisis money)
- AWS bill: 196 USD (September)
- 4.19-cip136 took 8 GPU-days to review
- Very much depends on the workload and model
  - Qwen 3.8 is dense, slower than Qwen 3.6 MoE



# What does it cost?

- Test rig: Hardware ~2400 EUR, ~750W electrical (pre memory-crisis money)
- AWS bill: 196 USD (September)
- 4.19-cip136 took 8 GPU-days to review
- Very much depends on the workload and model
  - Qwen 3.8 is dense, slower than Qwen 3.6 MoE
- **MUCH** (orders of magnitude!) cheaper than similar systems



# What does it cost?

- Test rig: Hardware ~2400 EUR, ~750W electrical (pre memory-crisis money)
- AWS bill: 196 USD (September)
- 4.19-cip136 took 8 GPU-days to review
- Very much depends on the workload and model
  - Qwen 3.8 is dense, slower than Qwen 3.6 MoE
- MUCH (orders of magnitude!) cheaper than similar systems
  - Sashiko: millions of tokens per review



# What does it cost?



- Test rig: Hardware ~2400 EUR, ~750W electrical (pre memory-crisis money)
- AWS bill: 196 USD (September)
- 4.19-cip136 took 8 GPU-days to review
- Very much depends on the workload and model
  - Qwen 3.8 is dense, slower than Qwen 3.6 MoE
- **MUCH** (orders of magnitude!) cheaper than similar systems
  - Sashiko: millions of tokens per review
  - SLTR: tens of thousands

# Latest Developments

# Latest Developments

- Existing: Reviews

# Latest Developments

- Existing: Reviews
  - "Does this break anything?"

# Latest Developments

- Existing: Reviews
  - "Does this break anything?"
- New: Triage

# Latest Developments

- Existing: Reviews
  - "Does this break anything?"
- New: Triage
  - "Do we actually need it in the first place?"

# Latest Developments

- Existing: Reviews
  - "Does this break anything?"
- New: Triage
  - "Do we actually need it in the first place?"
- Bleeding edge: Backports

# Latest Developments

- Existing: Reviews
  - "Does this break anything?"
- New: Triage
  - "Do we actually need it in the first place?"
- Bleeding edge: Backports
  - "How do we fit it in?"

## Code

- Got llama.cpp?
- Anything that can multiply numbers with at least 24GB of memory?
- Got commits you want to have analyzed?

Go to `https://github.com/uli/sltr` and give it a shot.

**Thank you for listening!**