文献阅读与撰写示例

王文浩

2024/3/14

提纲

- 学术论文的类型和结构
- 怎么读一篇学术论文
- 怎么写一篇学术论文
- 示例
- 附录：怎么写rebuttal

学术论文的类型

- 期刊论文、会议论文
  - CCF/IIE推荐国际学术会议、期刊目录

中国计算机学会推荐国际学术会议和期刊目录

（**2022** 年）

中国计算机学会

***标黄：表示现在2019年第五版目录中，升级至A/B
***红字：表示不在2019年第五版目录中， 直接申请A/B/C
***~~删掉~~：表示从2019年第五版目录中移除

https://www.ccf.org.cn/Academic_Evaluation/By_category/

学术论文的类型

- 原创论文（Original research/article）
  - 大多数论文的类型，具有原创性的发现
  - 具有严格的写作结构（稍后介绍）
- Review/SoK
  - 关于某个topic近期的研究回顾、总结
- 短文
  - Short paper/workshop paper
  - Poster：墙报展示，含正式出版或非出版物
  - Notes：关于某个特定技术的描述
  - Letter/Comment
    - 发表文章发表自己的观点意见，非正式论文
  - Communications：短期、快速（占坑）
  - Editorial：编辑起草或者指定专家撰写

原创学术论文

- 学术论文的基础：学术发现
  - 定义：解决的问题是什么？
  - 收集问题相关的信息和资源
  - 提出理论假设、技术方案
  - 实现、验证提出的方案，并收集数据
  - 分析数据、撰写论文

原创论文的结构

- Title
- Abstract
- Introduction
- Methods
- Results
- Discussion/Conclusions
- Acknowledgements
- References

原创论文的结构

Scientific process	Section of paper
Orienting readers	***Title***
What was done in a nutshell?	***Abstract***
What is the problem addressed?	***Introduction***
How did we solve the problem?	***Theory / Methods***
What did we find?	***Results***
What does it mean?	***Discussion***
What have we learned (in short)?	***Summary and conclusions***
Who helped us?	***Acknowledgements***
Whose previous work did we rely on?	***References***
Additional information	***Appendices***

提纲

- 学术论文的类型和结构
- 怎么读一篇学术论文
- 怎么写一篇学术论文
- 示例
- 附录：怎么写rebuttal

读论文的目的是什么

- When you read a research paper, your goal is to understand the scientific contributions the authors are making
- 当你阅读一篇研究论文时，你的目标是理解作者所做的科学贡献

怎么读一篇学术论文

- 先泛读：快速浏览全文
  - 阅读摘要：理解文章的主要发现以及它们的重要性。
  - 首先阅读全文以了解“整体画面”
  - 注意文中涉及的任何术语或技术
  - 记下不理解的任何问题或部分
  - 如果对文章中的任何关键概念不熟悉，求助搜索引擎。

怎么读一篇学术论文

- 再精读
  - 仔细重新阅读文章，特别是“方法”和“结果/结论”部分
  - 仔细检查图表、表格和图示
  - 在阅读标题和细节之前，尝试解释数据
  - 确保完全理解文章

读论文的几个原则

- Read critically:阅读研究论文必须是一个批判性的过程。不应假设作者总是正确的，相反，应持怀疑态度
- Read creatively:批判性地阅读一篇论文是容易的，而创造性地阅读则需要更困难、更积极的思考
  - It is always easier to tear something down than to build it up.

How to read a research paper. By Michael Mitzenmacher (Harvard)

读论文的几个原则

- 带着问题读：Critical reading involves asking appropriate questions.
  - 如果作者试图解决一个问题，他们是否解决了正确的问题？他们是否没有考虑到简单的解决方案？解决方案的局限性是什么（包括作者可能没有注意到或明确承认的局限性）？
  - 作者的假设是否合理？在假设的基础上，论文的逻辑是否清晰且合理，或者存在推理上的缺陷？
  - 如果作者提供数据，他们是否收集了正确的数据来支持他们的论点，并且他们是否以正确的方式收集了数据？他们是否以合理的方式解释了数据？是否有其他数据更具有说服力？

How to read a research paper. By Michael Mitzenmacher (Harvard)

读论文的几个原则

- 带着问题读：read creatively
  - 这篇论文中有哪些好的想法？
  - 这些想法是否有其他应用或扩展，作者可能没有考虑到？
  - 它们是否可以进一步通用到其它场合（场景）？
  - 是否存在可能的改进，可以产生重要的实际差异？
  - 如果你要从这篇论文开始进行研究，你将会做什么下一步？

How to read a research paper. By Michael Mitzenmacher (Harvard)

读论文的几个原则

- 适当做笔记
  - 如果有问题或批评意见，将它们写下来，以免忘记
  - 划出作者提出的关键观点
  - 标记最重要或看似有问题的数据
  - 这样的努力有助于第一次阅读论文，并在几个月后需要重新阅读论文时带来巨大回报

How to read a research paper. By Michael Mitzenmacher (Harvard)

怎么读一篇学术论文

- 写一篇关于该文章的“总结（summary）”
- 用自己的话描述文章-将文章提炼出其“科学本质”
- 记录“关键点”
  - 研究目的/提出的问题、假设、主要发现和结论、未解答的问题和任何意外之处

怎么读一篇学术论文

- Try to summarize the paper in one or two sentences
  - 几乎所有优秀的研究论文都试图回答一个具体的问题
  - 如果能简洁地描述一篇论文，可能你已经真正理解到了作者最初提出的问题和他们提供的答案
  - 一旦专注于主要观点，你可以回过头来尝试概述论文，以深入了解更具体的细节
  - 实际上，如果用一两句话总结论文很容易，那么请回过头来尝试通过总结主要观点的三个或四个最重要的子要点来深化你的概述

怎么读一篇学术论文

- 与相关工作进行对比
  - summary是一种尝试确定论文科学贡献的方法之一
  - 但要真正评估科学价值，必须将论文与该领域的其他作品进行比较。这些想法是否真正新颖，还是之前已经出现过？
- 值得一提的是，科学贡献可以采取许多形式
  - 有些论文提出新的想法
  - 有些实现了想法，并展示了它们的工作原理
  - 还有一些将以前的想法汇集起来，并将它们统一在一个新颖的框架下
  - 了解该领域的其他工作可以帮助你确定一篇论文实际上是做出了哪种贡献

我所期望的论文报告内容

- a one or two sentence summary of the paper.
- a deeper, more extensive outline of the main points of the paper
  - E.g., assumptions made, arguments presented, data analyzed, and conclusions drawn
- any limitations or extensions you see for the ideas in the paper
- your opinion of the paper; primarily, the quality of the ideas and its potential impact

提纲

- 学术论文的类型和结构
- 怎么读一篇学术论文
- 怎么写一篇学术论文
- 示例
- 附录：怎么写rebuttal

怎么写一篇学术论文

八股文

但把八股文写出彩需要很多科研锻炼

学术论文是一种什么文体

- 记叙文
- 说明文
- 议论文
- 散文、诗歌
- 小说
- ……

议论文

- 议论文是一种常见的文体，又叫说理文，可剖析事物、论述事理、发表意见、提出主张
- 议论文观点要明确，论据要充分，语言要精练，论证要合理，逻辑性要强。只有这样，才能做到以理服人
- 论点
  - 问题重要，方案可行，non trivial，效果好
- 论据
  - 用以支持论点的证据、理由、材料
  - 统计数据、实验数据、科学定理、参考文献
- 论证
  - 运用论据来证明论点的过程和方法

怎么写一篇学术论文

- 从一个好的框架开始
  - Skeleton、Outline
- 包括论文的章节标题、子标题以及关键段落的小标题
- 例如：introduction
  - Motivation/challenges/design/contributions

Universality of ac conduction in disordered solids

Jeppe C. Dyre and Thomas B. Schrøder
Department of Mathematics and Physics, Roskilde University, DK-4000 Roskilde, Denmark

标题

- 几乎是论文最容易被看到的、最重要的元素
- 搜索引擎的关键词
- extremely informative, simple and compact

摘要

- What was done and why?
- How it was done
- What were the main results
- What are the main conclusion

- An abstract must be understandable without reference to the rest of the paper
- 不允许出现文章中不包括的内容、对图表和其它参考文献的引用、所提出方法的技术细节

摘要

An abstract summarizes, in one paragraph the major aspects of the entire paper in the following sequence:

1. *The question you investigated* (**Introduction**)
   - Clearly state the purpose in the first or second sentence
2. *The experimental design and methods used* (**Methods**)
   - clearly express the basic design of the study
   - briefly describe the basic methodology used (without detail), indicate key techniques used
3. *Major findings, key quantitative results or trends* (**Results**)
   - report results relevant to the questions asked
   - identify trends, relative change or differences
4. *A brief summary of your conclusions* (**Discussion**)
   - clearly state the implications of the results

引言

- 引言提供了足够的背景，使读者能够理解研究，而无需参考先前的文献，并为研究提供了合理的依据
- A good introduction:
  - Presents the scope of the problem
  - Reviews related literature
  - Sets the stage for the methods chosen
  - Ends with clear objectives (sometimes paper organization)

方案、设计、实现

- 这项研究具体是如何开展的
- 提供细节，使得读者能够：
  - 评估方法的可行性、价值
  - 评估结果的合理性
  - 重复研究的过程
- Justify your choice of methods
  - 其它方法可以被引用，但不需要过多篇幅进行介绍

评估、结果

- 列举实验结果，并验证论文所提出的科学假设（方案是否合理、问题是否得到解决）
- 主要列举支撑重要结论的数据（去除无关数据）
- 不要选择性地列举数据、选择数据
- 多使用图、表
- 通常使用过去时态

讨论、相关工作

- Generalize based on your results
  - 其它平台、其它场景、防御措施
- Point to exceptions and inconsistencies, limitations
- Discuss shortcomings and open issues
- Relate your work to previous studies

- 避免
  - 重复已经阐述过的方案、结果
  - 讨论不重要的发现（使得reviewer忽略了重要的发现）

Acknowledgements

- 资助信息
- 帮助完成或完善该研究论文的人、集体

图表

- 所有的符号、字母、维度都有相应的label
- 需要有清楚的标题，对图表进行适当解释
- 图表内容在黑白情况下是可读的
- 图表必须在正文适当的地方被引用
- 引用时，需要从图表得出结论，而不是简单地重复图表的内容

怎么写一篇学术论文

- Simplicity and clarity in writing
- All words used in scientific writing should be:
  - Simple
  - Essential
  - Specific
  - Familiar
- Avoid redundancy(避免重复的句式)
- 想象读者对论文topic几乎一无所知
  - 所有用到的术语都要定义并且清晰、无歧义地呈现

怎么写一篇学术论文

- 写作的关键

- 逻辑性、论证要严密，不要跳跃，不能认为显而易见而不写

怎么写一篇学术论文

- 要写清楚关键步骤，详细写

- 但不要事无巨细，写无关紧要的细节
  - 不是所有你做的东西都写上去，我们不是记叙文也不是说明文

提纲

- 学术论文的类型和结构
- 怎么读一篇学术论文
- 怎么写一篇学术论文
- 示例
- 附录：怎么写rebuttal

示例-标题

vSGX: Virtualizing SGX Enclaves on AMD SEV

Shixuan Zhao*, Mengyuan Li*, Yinqian Zhang†‡✉, Zhiqiang Lin*✉

*Department of Computer Science and Engineering, The Ohio State University
†Research Institute of Trust-worthy Autonomous Systems, Southern University of Science and Technology
‡Department of Computer Science and Engineering, Southern University of Science and Technology

示例-摘要

- 背景
- 问题
- 解决方案
- 核心贡献
- 实现和评估

***Abstract*—The growing need of trusted execution environment (TEE) has boomed the development of hardware enclaves. However, current TEEs and their applications are tightly bound to the hardware implementation, hindering their compatibility across different platforms. This paper presents vSGX, a novel system to virtualize the execution of an Intel SGX enclave atop AMD SEV. The key idea is to interpose the execution of enclave instructions transparently to support the SGX ISA extensions, consolidate encrypted virtual memory of separated SEV virtual machines to create a single virtualized SGX-like address space, and provide attestations for the authenticity of the TEE and the integrity of enclave software with a trust chain rooted in the SEV hardware. By design, vSGX achieves a comparable level of security guarantees on SEV as that on Intel SGX. We have implemented vSGX and demonstrated it imposes reasonable performance overhead for SGX enclave execution.**

示例-引言

• 交代背景

Over the past few years, we have witnessed a tremendous growth of the use of trusted execution environments (TEEs), such as Intel Software Guard Execution (SGX) and AMD Secure Encrypted Virtualization (SEV). TEEs have great promise in protecting both confidentiality and integrity of program code and data from malicious system software and operators, which are extremely valuable for clouds where the computing platform is not fully trusted by its customers. Existing cloud deployment of TEEs includes Alibaba Cloud's SGX VM instances [7], Microsoft Azure's confidential computing [3], Google's confidential virtual machines [4], and so on.

As a prominent TEE platform from Intel, a dominating player in the general-purpose CPU market, SGX has once become the de facto standard for building TEE-based applications. A rich ecosystem with abundant open source projects and commercial products has been built atop SGX, including SGX-based password manager [40], SGX-based anonymity network [38], privacy-preserving data analytics (e.g., [57], [60]) and machine learning (e.g., [41], [50]), SGX-based game protection (e.g., [16], [54]), privacy-preserving contact-tracing (e.g., SafeTrace [1]) and blockchains [19] using SGX, and SGX-based IoT network [48], etc.

However, the ISA extension of SGX mandates a clear separation of software applications into trusted and untrusted components, such that the trusted software components are

• 提出问题

Decoupling TEE software applications from the underlying TEE hardware is a strong desire of the cloud providers. For instance, Google's Asylo project [8] aims to provide a unified SDK interface so that the same TEE source code developed with Asylo can be compiled and run on any TEE hardware; Amazon's Nitro Enclaves [5] use virtualization technology to form secure enclaves, so that confidential workloads can run without SGX. However, neither of these methods can achieve binary compatibility. Ideally, the cloud providers would offer their customers an option to build their applications once, in accordance with the SGX semantics, for instance, given the large volume of existing SGX-based projects, and deploy them on a variety of cloud servers, which may or may not have the hardware capabilities of SGX.

Moreover, the customers should be provided the freedom of choosing the level of trust they have on the cloud providers. For instance, for users who fully trust the cloud providers, hypervisor-based enclaves (e.g., Nitro Enclaves [5]) can be used. But for other users who do not, either SGX or SEV can be chosen from two different levels of trust: SGX features small user-space enclaves with all other software components exposed to the untrusted hypervisor, while SEV protects the entire VM and allows flexible deployment of existing applications, at the cost of a larger attack surface. However, to the best of our knowledge, there is no technique that could combine the benefit of both SGX and SEV so that a user can enjoy SEV-protected VMs for a private computation environment while still be able to run existing SGX enclave binaries.

示例-引言

- 两句话介绍核心贡献

To demonstrate such a feasibility and practicality, in this paper, we present vSGX, a system that provides binary code compatibility of partitioned SGX enclave software and enables its direct execution atop AMD SEV. Conceptually, vSGX can be considered as an SGX hardware module that is plugged into an SEV machine. The key idea behind vSGX is to leverage the VM protection provided by SEV, and execute trusted enclave of a legacy SGX application in a separated

- 核心设计

be considered as an SGX hardware module that is plugged into an SEV machine. The key idea behind vSGX is to leverage the VM protection provided by SEV, and execute trusted enclave of a legacy SGX application in a separated

示例-引言

- 挑战

While the idea of virtualizing SGX enclaves using SEV might appear to be simple, it in fact faces many non-trivial challenges (§III). These challenges include how to interpose the execution of enclave instructions in AMD SEV; how to handle enclave entrance and exit since with vSGX an enclave is executed in a separate EVM; how to handle cross memory access between the EVMs and AVMs; how to deal with the untrusted code in the AVM's OS or even a malicious hypervisor; and how to perform SGX remote attestation on AMD machines. We have fortunately addressed these challenges when designing vSGX (§IV).

示例-引言

- 实现和评估

We have analyzed the security of vSGX and discussed that our design has achieved a comparable level of security as with Intel SGX, through the use of the primitives provided by AMD SEV (§V). We have also evaluated its performance overhead with a set of benchmarks and real world applications (§VI). Our experimental results from the benchmarks show that while many of the enclave instruction executions (particularly `EENTER` and `EEXIT`) are indeed slower when running in SEV compared to running in Intel CPU, these overheads will only be observed by ECall or I/O intensive applications. Our evaluation with real world SGX applications shows that the overhead of vSGX is reasonable. Therefore, we believe vSGX represents a practical way of executing SGX enclaves atop AMD SEV.

示例-引言

- 总结

示例-背景知识

- 看论文

示例-系统设计

- 总体设计（Overview）

示例-详细设计

The architecture of vSGX is illustrated in Figure 1. There are five components inside vSGX: (1) *Instruction Emulation* (§IV-A), (2) *Enclave Manager* (§IV-B), (3) *Memory Management* (§IV-C), (4) *Cross-VM Communication* (§IV-D), (5) *Remote Attestation* (§IV-E). In this section, we present the detailed design of these components.

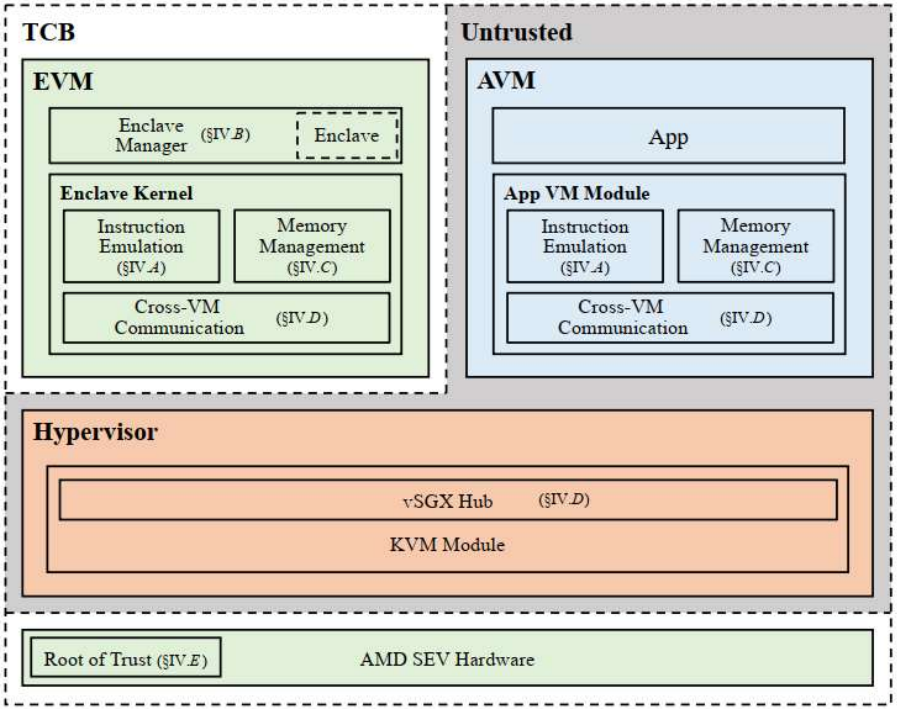

Fig. 1: The vSGX architecture.

示例–安全性分析

V. Security Analysis

In this section, we analyze the security of vSGX and discuss how vSGX achieves the desired security goals, namely our **G2** and **G3**.

- 系统性地分析所提出方案面临的攻击面，证明达到目标的安全性

示例-评估

emulation packet, a switchless-syncing packet and a fetch-and-map packet.

- An instruction-emulation packet is dispatched to its corresponding enclave as specified by the EPC page it operates on. The verification is enforced in the local dispatcher of that enclave according to the Intel SGX specification as discussed in §V-A.
- A switchless-syncing packet is first dispatched to its corresponding enclave. Then, the enclave will check its switchless-syncing list to see if the page to be synced is in the list. If and only if so, the packet is accepted. The

VI. Evaluation

We have implemented vSGX with 16,167 lines of C code (LoC) and 121 lines of x86-64 assembly. The AVM module contains 6,377 LoC and 121 lines of assembly, 8,840 LoC in the enclave kernel, 250 LoC for then enclave manager and 700 LoC in the hypervisor's KVM module. The source code of vSGX is made available at github.com/OSUSeclab/vSGX. In this section, we present the evaluation result. Since we have answered the security questions of vSGX in §V, in this section we would like to answer the questions related to the performance overhead of vSGX. To this end, we designed

and chose a set of benchmarks and real world applications to understand the overhead at both the component level and the application level. A set of microbenchmaks were designed and reported in §VI-A1 to reveal the performance on an instruction and component level; A macrobenchmark software was chosen in §VI-A2 to reflect overall performance. Finally, we also report the compatibility and performance overhead for real world SGX application in §VI-B.

	Leaf	Average Overhead (μs)	Packets Sent
ENCLS	EADD	1421.23	3
	EAUG	990.20	2
	EBLOCK	840.85	2
	ECREATE	3719.06	3
	EDBGRD	N/A	N/A
	EDBGWR	N/A	N/A
	EEXTEND	986.76	2
	EINIT	811.03	2
	ELDB/ELDU	1958.13	4
	EMODPR	1071.26	2
	EMODT	976.15	2

示例-讨论

VII. Limitations and Future Work

vSGX can be improved in multiple avenues. We list some of the ideas to improve vSGX below.

- vSGX currently does not support enclave debugging. Specifically, `EDBGRD` and `EDBGWR` are not supported. We leave the support for enclave debugging to future work.
- The cross-VM memory syncing in vSGX cannot reflect real-time changes. As such, memory barriers and atomic instructions between the enclave and the application code will not behave correctly. This can interfere with locks implemented with atomic instructions sharing with the untrusted world. A solution is to use OCalls to implement locks on shared memory pages.
- vSGX does not yet fully support Intel's `CPUID` semantics. For instance, if the software uses `CPUID` to check if SGX is supported, the check would return negative. This can be supported by software emulation, but as the behavior of `CPUID` is architecture-dependent, we leave it to future

示例-相关工作、结论、致谢

VIII. Related Works

There are numerous efforts in supporting the growth of the TEE software developer community. In particular, there are a variety of SDKs (e.g., Intel SGX SDK, Rust SGX SDK [69]). Efforts have been made to provide uniform TEE A
to the developer regardless of TEE implementa
ples include the Asylo framework proposed by
the Open Enclave framework by Microsoft [
Open Portable Trusted Execution Environment (C
There are also approaches of running legacy co
an enclave as demonstrated in SCONE [12], Ha
Graphene-SGX [21]. Others aim to integrate S
and containers [9], [62], [63] and to support er
tion [28], [53]. In this work, we focus on prov
compatibility of SGX enclave applications and d
the execution of SGX enclaves on AMD platfor

IX. Conclusion

We have presented vSGX, a novel system to virtualize the execution of Intel SGX enclave atop AMD SEV. With transparent instruction emulation, cross-VM memory synchronization, and tight integration with the SEV-based memory encryption and isolation, vSGX provides binary-compatible support for SGX enclave applications without losing security. We have implemented vSGX and demonstrated it incurs reasonable performance overhead for SGX applications.

Acknowledgments

We thank the anonymous reviewers for their insightful comments, which have significantly improved the paper. Shixuan Zhao and Zhiqiang Lin were partially supported by NSF grant 1834213 and 1834216, and Yinqian Zhang was in part supported by Ant Group.

总结

- 学术论文的类型和结构
- 怎么读一篇学术论文
- 怎么写一篇学术论文
- 示例

- 联系方式：
- wangwenhao@iie.ac.cn

提纲

- 学术论文的类型和结构
- 怎么读一篇学术论文
- 怎么写一篇学术论文
- 示例
- 附录：怎么写rebuttal

附录：怎么写rebuttal

- How we write rebuttals. by Devi Parikh, Dhruv Batra, Stefan Lee

- 主要面向会议论文
- 期刊论文的major revision: response to reviewer comments，也是类似的

学术会议的审稿流程

- 想象中
  - 学者们手头充裕，聚集在一个安静的空间里，仔细研究他们工作的细微差别
  - they imagine scholars, with plenty of time on their hands, gathered around a quiet space carefully pouring over the nuances of their work
- 现实：a large crowded marketplace
  - 一种混乱而杂乱无章的情况；参与过程的不同角色都在努力实现不同的目标，每个人在有限的时间和注意力范围内工作，面对不完全的信息，同时还要应对多项责任

Rebuttal

- 对象
  - Reviewers：他们可能以不同程度阅读过你的论文，但可能已经忘记了其中的一些细节，或者一开始就没有理解它们
  - 以及PC(AC) Chair：可能对你的工作了解更少，一个良好的指导原则是假设他们只会阅读reviews和rebuttal

Rebuttal

- 目标
  - Reviewers: 澄清疑惑，回答问题，纠正误解，对错误描述提出反驳，并诚心努力将反馈融入并改进您的工作
  - 以及PC(AC) Chair: 说服他们相信你已经真诚努力，提供对review comment的摘要，帮助他们理解是否解决了审稿人的关切，揭示不诚信的审稿行为，并最终帮助他们做出决策
- rebuttal should be thorough, direct, and easy for the Reviewers and Area Chair (RACs) to follow

步骤

- Itemize reviewer comments. 整理每位审稿人提出的个别评论、问题和关切。将所有内容放在同一个地方有助于识别共同关注点，并避免疏漏。尽快完成这个步骤，以便及早确定是否需要进行新的实验或分析（如果允许）
- Brain dump possible responses. 逐条回复审稿人的每个comment，以简略的文字记录你的想法，不需要考虑风格或长度。具有说服力和简洁性是一个减法过程
- Write a draft rebuttal：将观点转化为具体的rebuttal，写作时要简明扼要，不用担心篇幅。涵盖每个观点，并稍后进行修整和优先排序。
- Review and revise. 重新阅读审稿意见和rebuttal，确保所有问题都得到了解决。优先处理重要的关切点，并开始努力满足篇幅限制

tips

- 尽量能够体现论文的正面价值
- 按问题的重要性排序
- Consolidate common concerns.
  - Save space by responding to multiple reviewers at once if they share related concerns.

@R2,R3 – Why gradient-based explanations? Why not align attention to human importance? Gradient explanations directly link model decisions to input regions (being decision gradients after all) and so aligning these importances ensures the model is basing its decision on human-attended regions. Further, gradient-based explanations are a function of all the parameters of the network (L443-447). In contrast, attention is a bottom-up computation that relies only on the image and the question (L483). Even with appropriate attention, the remaining network layers may still disregard the visual signal in the presence of strong biases in the dataset.

tips

- 正面、直接回答reviewer的问题，然后提供细节
  - “Are these averaged across multiple runs?”:
    - “Yes, we averaged across 5 random seeds.”
  - “Are the segmentation masks used during training?”:
    - “No, they are only used to evaluate our results.”
- Respond to the intent of the questions（回应问题的意图而不是问题本身）
  - Why didn’t you evaluate on GLORP3?” may generally be calling your experiments into question.
    - Answer, but then point out that you’ve already evaluated on X,Y, and Z which should be sufficient

tips

- Keep things self-contained.
  - Reviewer可能对你的论文记忆不深，而且他们可能不会再详细阅读它。
  - 重新介绍任何缩写词，提醒他们实验设置的相关细节。
  - 请注意，论文的回应应该使得即使对这些论文不熟悉的人也是讲得通的、有意义的。
- get credit for details you already included
  - 如果审稿人提问的问题在原文中有答案，请指出来
  - Xx章节、xx图表，然后再重复一遍

tips

- Don’t promise, do
  - 不要说“We will discuss Singh et al. in the paper.”，在rebuttal中就提供一个discussion
  - 不要说“We will explain what D_{RT} stands for in the paper”，在rebuttal中解释它代表什么
  - 然后说明，会加到最终版的论文里

tips

- Be transparent. 指出来：
  - Reviewers要求补充实验，但是该会议不允许rebuttal阶段补充实验数据？
  - 没有足够的算力完成reviewer要求的实验？
- Don’t forget the humans on the other end
  - Typo list? Thank you. Pointers to relevant work? Thank you. Detailed musings about future work? Thank you. Add at least a short blurb acknowledging these things!
  - Finding points where you do agree with the reviewer and acknowledging them can help with the latter.