Writing a System Security Paper

(A Checklist)

Getting your paper accepted in one of the top venues is hard, no matter how clever your idea or brilliant the implementation. Clearly, there is no such thing as a simple recipe for writing a paper that you can follow to produce a great paper, but there are some do's and don'ts that you should try to follow. Make sure that you have considered *all* of these points before submitting your paper.

0. General

Style/presentation

- Omit needless words (you will find that most adjectives are superfluous).
- Be positive - use positive wording (not "impossible", "bad", "infeasible")
- Avoid the passive form!
- Avoid extremely defensive writing. Present your arguments in the Introduction, Background, etc. and just focus on the solution (no more "we argue that…") from the Overview on.
- Use topic sentences
- While you are at it: read Strunk & White (The Elements of Style): https://faculty.washington.edu/heagerty/Courses/b572/public/StrunkWhite.pdf
- Read Strunk & White again; repeat
- Do not use complicated formalisation unless absolutely needed (plain English is much easier to read)
- Skim the paper to see if there is too much repetition. Readers find that very annoying. For instance, it is okay to say something like "To the best of our knowledge, we are the first to do X"— once. Saying it twice is not necessary. Three times is very, very annoying.
- What is a good idea, however, is to start with a helicopter view of your solution and keep zooming in as you get further into the paper, refining the interesting points. So, explain the (entire) big picture early, then fill in the boxes in the design section and then perhaps even more in the implementation, etc. Essentially that is also telling the same thing multiple times, but now with increasing amount of detail.
- Pay attention when citing papers. You want to mention the authors ("Peter and Dorothy Denning et al. were the first to introduce information flow tracking [1]"), or the project ("SystemX [2] was the first to do X, but it could not do Y"). Probably the only legitimate reason for not mentioning either is that you are making a general statement and not

discussing a specific research project (“While there are many solutions that build on symbolic execution [3,4,5,6], very few of them scale to large programs”).
- Use footnotes sparingly[1]
- Let us agree that we write American English. It is neither better nor worse than British English, but it helps if we all pick the same. Also AE is more familiar to the reviewers (on several occasions, I received reviews of papers in British English with a list of “language corrections”).
- Often multiple people work on a paper, so go through the paper to check for consistency (in terminology, but also in argument)

Message
- Present one idea per paper only. Two ideas are harder to sell, even if they are good ideas.
- Use running examples when possible.
- Avoid sounding boastful, pedantic, preachy, or colloquial. You probably want to avoid humour also. You are not writing for the school newspaper.
- What have you learned from the work? Spell out the lessons explicitly
- Know your audience. For the submission, the audience is the program committee, not the final audience!

1. Abstract:

- An abstract is a replacement for the entire paper. Ideally, it’s the paper in an extremely compressed (but nearly lossless) form. Keep it short, comprehensive, and to the point.
- There is no perfect recipe, but starting with the abstract (and title) is usually a great idea. It forces you to think about the problem, angle, contributions, terminology, key results and make sure you have all the right ingredients lined up for the paper. If you’re working with others, it’s also a great way to make sure everybody is on the same page.
- Writing a good abstract is not trivial. But remember: if you can’t explain it in a few sentences, you don’t understand it well enough.

2. Introduction:

The introduction may be the most important section. Make sure you spend significant time on this. Here are some suggestions:

- Keep it short! Just explain the general idea and then explicitly list the contributions. Save all the convoluted arguments for later sections. It is very annoying to have to read 3 or 4 pages before you find the actual contributions. Make sure to have them on p2 at the latest.

[1] they break the flow

- In fact, you may want to give an ultra-short summary of what the paper contributes in the first or second paragraph.
- Dive right in. Exaggerating slightly: here is no need to explain the history of computing. Come to the point immediately. (See also the points above.)
- Also, while you should sketch the context of the work, do not mistake the introduction for a related work section. Even so, you *may* want to touch upon the most directly related approaches very briefly (e.g., 2 sentences) to emphasise how different your solution is.
- Keep it simple. You want it to be appealing and that will never work if the reader does not understand half of it.
- Make sure to show that you solve a really important problem (hint: this is often easy if you can prove real-world impact backed up by concrete numbers). Otherwise the reviewer will think: why bother? Also argue that you solve it well, ideally by previewing the results (suggesting that things are really *much* better now).
- Rephrasing the above: you believe in the paper (otherwise, don't submit), so make sure to argue for it. Why should the PC accept it?
- Related: make sure the reviewer understands what is *novel* about your work. For instance, if you manage to stop twice as many attacks, but you cannot explain clearly what is novel about the approach, it will be hard to get the paper accepted. Tell the reviewer explicitly!
- Even so, keep your language modest. If you are too boastful, reviewers will see it as a challenge to find some reason to reject your paper.
- Be upfront about the things your solution cannot do. If reviewers notice that you overclaim or find out on page 12 that your solution works only in a few cases, they will hate the paper. If you scope your paper such that you address specifically these cases, this is often less so ("expectation management").
- Mention your contributions *explicitly*. This is what the reviewers will look for when checking your evaluation section and when discussing your paper. Do not list 10 contributions. Pick the things you want reviewers to discuss.
- If you think you have single contribution ("I've built a system that does X"), try to split it up for convenience (the 3-contribution design/implementation/evaluation split is common).
- If possible/appropriate, mention that you will make the source code available when the paper is accepted (and then do so!). PCs like working tools to build on.

3. Threat model and assumptions

- All papers on defenses *need* a threat model.
- Convince the reader that the threat model is realistic.
- All attack papers require listing the assumptions.

4. Overview

- How would you write about the end-to-end design from a high-level perspective in <=1 page? This is the Overview section (Abstract : paper ~= Overview : Design).
- Try to write it from a user workflow perspective. For example, how would users use your system? What is the input and expected output?
- Starting from user input, describe the flow through the components of your system. List all the components (and names) and mention their high-level responsibilities.
- Try to map every design subsection into a component (e.g., if you do optimizations, consider having a “Optimizer” component), so it’s much easier to write a simple and comprehensive Overview.
- Include an Overview figure.

5. Implementation

- Go low on implementation details.
- If you have both design and implementation, again, make sure there is not too much repetition. Refinement is what you want, not repetition. Fill in the boxes with implementation details where needed for the understanding of the paper (e.g., where there may be doubts as to how you could implement this), or where this explains some of the results.

6. Evaluation

- Make sure your evaluation backs up your claims. (Really, check that they do!)
- Make sure every new claim/contribution you make is backed up by experimental results.
- Explain all plots and all the results. Tell the reader what he/she should notice. Explain outliers, and false positives/negatives, etc. Do this in a critical way and make sure the evaluation converges. In other words, summarize what we learned from the experiments and show our the results confirm all your claims (you can typically do this at the end of every subsection in the Evaluation).
- Familiarise yourself with Gernot’s benchmarking crimes:
  https://www.cse.unsw.edu.au/~gernot/benchmarking-crimes.html
  Do not do any of this.
- If you do malware experiments: read the Prudent Practices paper:
  http://www.cs.vu.nl/~herbertb/papers/
- Do not use 10 year old datasets, or 10-year old exploits, etc. Try to find something recent.
- Compare your results with competing solutions as much as possible, but do so in a fair way (e.g., SPEC results across the same configuration, benchmarks, etc.).

7. Related work

- Related work can either be at the beginning (say, after the intro), or at the end (after the results). Both are fine, but there is a subtle difference in what they convey. For instance, related work in the beginning helps if your paper requires certain background knowledge in the remainder of the paper, if you need to distinguish yourself from other approaches early (especially when the distinction is not immediately obvious), or if you need to argue that contrary to what people tend to think, existing solutions are not solving the problem. In all the other cases, related work at the end is normally a better idea, since it allows you to place the “real meat” earlier in the paper (keeping the distance between Abstract and Overview short is important).
- Treat related work fairly. Being negative about related work is not what you need to do and often works counter productive. Factually point out limitations and show where your solution differs.
- You may not be able to cover all possible related work and often you don’t have to. Make sure that you show that you are aware of the important work in the area. Often you can summarise a score of projects by referring to an “approach”, rather than individual papers (“Many research projects try to detect information theft by means of dynamic taint analysis [12,13,14,15] and are thus susceptible to taint laundering by malicious code [16]”).
- Make sure to check whether anyone on the PC has published (even slightly) related work and if so, make sure to discuss it!.
- Prefer system names (e.g., Howard [14] relies on X to perform Y) over author names

8. Limitations / discussion

- Your solution is not the final answer. List the limitations. It is much better that you do this than the reviewers. In fact, listing it yourself makes it harder for the reviewer to reject your paper because of it. (See also “expectation management”).
- Do not sweep anything under the carpet. Some reviewer will probably find out and this will taint the paper (not just for this submission, but also for subsequent ones).
- Placing the limitations/discussions section in the right place is tricky. If the limitations are only implementation-related, you may want to have a “Limitations” paragraph or so within the Implementation section. More fundamental limitations should be discussed in a separate “Limitations” section, typically before the Evaluation.

9. Common Language bugs

- e.g., i.e., (with comma)
- therefore (you probably did not mean therefor)
- don’t, won’t can’t → do not, will not, cannot → do not make your text colloquial

- try not to split infinitives (while this is no longer a hard and fast rule, to boldly split infinitives is considered ugly)

10. Log (everything)

- Make sure everything is reproducible. Try to keep your code, the data, the paper and all scripts to produce plots in the same repository. If the size of the data is too large, use one of the large storage space provided by the university or SURFnet.

11. Figures

- Make sure they look good in black & white also
- Make sure fonts are readable
- Try to make them attractive

12. Layout

- Remove ugly widows and orphans (hint: \looseness=-1 may help).
- Make sure section headings are not at the bottom of the page
- Balance sections and subsections properly (e.g., no section should have only a single subsection).
- Place figures/tables at the top of the page whenever possible
- Space saving tips:
    - Reduce bibliography size (e.g., footnotesize)
    - Add negative vspace (e.g., \vspace{-0.3cm}) below the caption of tables and figures.
    - Turn subsections into named paragraphs.
    - Use \small tables and smaller figures whenever possible.

XX. Parting shots

Finally, we have a reputation that we produce solid stuff and you can trust our results and we really want to keep that. So do not submit anything that you *know* will not work. Also, no papers please about security mechanisms that may today's attacks, but can be trivially circumvented by attackers. It goes without saying that cheating is absolutely unacceptable. It is okay, however, to show your solution (in which you believe) in a positive light. Don't worry the reviewers will put it in a bad light anyway.

Resources

Latex MSc template:
https://www.vusec.net/download/?t=docs/latex/msc-thesis-template.tar.bz2