Consolidated Review of *Revealing Middlebox Interference with Tracebox*

1. Strengths:

The paper is well motivated, and I can imagine using it. Paths are complicated, middleboxes can cause various behaviors, and *traceroute* is widely used: A *traceroute*-like tool to expose the complexity and middlebox behavior would be very useful. The initial steps towards a tool, which is capable of revealing middleboxes on a path, are promising.

The tool seems quite cool. Tool is available (and includes unit tests!). The tool seems to bring together flexibility in probing---via both setting header fields and using more than one packet (often key). Unlike some earlier work on middlebox discovery, only needs control of the source. Additionally, the reporting of what is going on seems cogent and clean. The tool seems like a good basis for both operational debugging and developing understanding about middlebox prevalence and behavior in the research community.

2. Weaknesses

The tool itself is a small incremental improvement of *traceroute*.

It is unclear how many middleboxes this tool actually misses (especially stateful ones). The location inference results are weak due to the dependence on RFC1812 compliancy.

Could be clearer on what the tool does automatically and what you have to tweak, configure, or hand run.

The measurement study is limited. The correctness of *tracebox* has not been validated in controlled experiments (in the wild, or with real hardware). Validation currently involves only detecting click elements the authors developed. What are your false-negatives?

Could be clearer on the limitations of the tool. Does not detect any middlebox behaviors that first require the 3-way handshake to be completed (e.g., mangling/hijacking BitTorrent messages to throttle or stop BT usage)

While the paper does a very workmanlike job at developing *tracebox* and producing a nice tool, the techniques aren't particularly novel. The technique of using packet quotations is what *traceroute* has always been based on. The authors just use more of the packet than the traditional *traceroute*. Further, Malone and Luckie used the same sorts of techniques in "Analysis of ICMP Quotations" in PAM 2007.

3. Comments

Tracebox offers improvements over *traceroute*. The tool provides more freedom with respect to the header fields of the probes and can check whether these header fields changed (only partially, unless router is RFC1812-compliant). This results in some significant limitations.

Intuitively a provider would place a middlebox close to the endpoint (e.g. the last hop before the destination). If this node is not RFC1812-compliant, you can miss many changes made by that middlebox (since there is no RFC1812-compliant router after that one). Also, you might actually miss all modifications if the node checks the TTL value first and returns the ICMP error message before making any changes. It seems like, in terms of debugging problems, understanding behavior near the destination would be very important.

A substantial fraction of the middleboxes could be located very close to the source (e.g. first hop is a firewall) but since you need the response from a compliant router these boxes would likely be placed in the core (based the router distribution shown in figure 2b).

How did you perform the normalization for Fig 2b? Most compliant routers seem to be in the core, but I'm not sure I buy your argument that that is of the highest importance - you'll miss any changes past that point. Presumably, the vast number of possible edge networks where we don't have VPs are the places we'd most like to know about middleboxes.

Where is the first / last RFC1812-compliant node on a path? For any nodes after the last compliant one you are going to miss most header modifications. It would be nice to have a non-normalized graph of the distribution of # hops from source to first compliant router and # hops from last compliant router to destination. Given that RFC1812 is almost 20 years old, what hope do we have for greater deployment? It would be neat to include a longitudinal study of adoption.

I didn't understand why 3.2 is a use case. Isn't it more of an assessment of coverage?

The most important deficit: It is not clear how the tool would detect any interaction with stateful middleboxes (e.g. drop of out-of-window packets, ACK rewriting, payload rewriting, etc.). Honda et al. have a solution for this when the user has control over both server and client. In 3.3, you talk about opening up a TCP connection, but the description of the tool in 2 doesn't make it clear if this is part of the tool.

The results about inferring the approximate location of a middlebox are unclear. Figure 4b uses a normalized distance, whereas 4c describes absolute distances. It is not possible to draw conclusions about the accuracy in 4b because of this (should either use absolute or normalized distances for both plots). In Fig 4c, how many VPs contribute to the large steps at hop 4 and 5? How many paths are even longer than 13 hops?

In Fig 4d, how do you know it was a middlebox and not the server itself? Are you not including probes that actually reach the target? Unclear how to read Fig 4d, given that each VP performed its own DNS resolution, so might be probing a different target.

Section 4.1 is somewhat out of place. The rest of the paper talks about middlebox detection whereas this one anecdotally describes a client configuration problem.

4.2: Which destinations was the proxy operating for?

4.2: While the loop might not be ideal, is it that big of a deal if the destination only wants to receive port 80 traffic? It's operating as a poorly setup firewall, in effect.

As you note, PlanetLab is not the most interesting testbed for studying middleboxes. Could you instead deploy on something like Project BISMark? Why did you only use 72 PlanetLab sites? How did you pick them?

I am sort of on the fence about how to balance the strength of the nicely developed tool with the weakness of a fairly thin methodological contribution. It seems to me that this paper's fate depends on the submission pool and where the bar will ultimately fall. I liked the range of problems **was** able to find. By only real concern with *tracebox* is with its (lack of) validation. Currently, the authors were able to detect the click software elements the authors built; while that shows that *tracebox* doesn't have obvious bugs, that doesn't establish the degree of accuracy of the results *tracebox* generates in the wild.

Middleboxes have been known to do all kinds of weird stuff (incl. mangling quoted packets inside the ICMP). It should be easy to validate *tracebox* by subjecting a bank of (off-the-shelf) middleboxes to it, and manually validating the outcomes. Alternatively, it would be great if the PIs of the Planetlab nodes *tracebox* found to be behind unsavory middleboxes could validate whether they indeed are behind middleboxes with the inferred behavior. Without some real-world validation it is hard to take *tracebox*'s results at face value.

4. Summary from PC Discussion

This is flagged as quick accept, but to post a short summary as discussion lead:

I think we generally all agree on strengths/weaknesses: a useful tool, although one with limitations and little technical novelty.

Strengths:
- ❖ Important problem
- ❖ Useful tool that they make available (Mark: tool is here http://www.tracebox.org/)
- ❖ Various reviewers had more specific strengths, but they mostly boil down to the two above

Weaknesses:
- ❖ Little technical contribution
- ❖ Lack of validation outside of unit tests
- ❖ While useful in some cases, limited in terms of what middlebox behavior it can uncover (only stateless) and in terms of localization
- ❖ Paper is unclear on limitations, unclear on to what degree the tool works automatically vs. needing a lot of tweaking of parameters to uncover behavior
- ❖ Small measurement study is interesting to read but not conclusive about general middlebox behavior (preliminary, small, limited to PlanetLab

5. Authors' Response

We are graceful to anonymous reviewers for their relevant feedback. We took most of their comments into account for the camera ready version of our paper.

One of the reviewers' key points is about the lack of clarity on tracebox limitations. We tackle this in the camera ready version of the paper, specially in Sec. 2. We also provide deeper explanations on tracebox and how it works (Sec. 2). Sec. 3 has been renamed ("Validation & Use Cases") to better reflect its content. Sec. 3.3. ("TCP Sequence Number Interference") has been strongly improved in terms of writing and results.

One of the reviewer pointed out the uselessness of Sec. 4.1. We think the result discussed in Sec. 4.1 is interesting as all of the websites were based in China which makes us believe that the interference was due to a middlebox sold by a Chinese manufacturer. Moreover, our result shows the implication on new TCP extensions as they can be impacted by such a behavior. We clarify discussions in Sec. 4.1.

Finally, it is worth to notice that this paper does not aim at providing a complete bestiary of middleboxes and their behavior. Rather, this paper aims at presenting tracebox and how it could be used by researchers and operators to identify middleboxes along a path and, possibly, to debug strange behavior. Next steps should improve our understanding of middleboxes (behavior, types, ...) by largely deploying tracebox. For instance, we are currently discussing with BISMark to deploy tracebox in their measurement infrastructure (as well as SamKnows). We are also deploying tracebox in IPv6 networks.