ABSTRACT
Documentation and collaboration are critical elements for malware analysis, however current tool support falls short of enabling a means for these activities in a low level domain. An initial survey not only confirms how important these elements are, but also the ways in which they are not currently supported. We have built a prototype that provides collaborative documentation through comments over shared comprehension artifacts. A preliminary evaluation, through interviews with users, shows how this approach holds promise and provides many avenues for a more social approach to reverse engineering even within this low level, hostile domain.

ACM Classification Keywords
D.2.7.c Software Engineering: Documentation; D.2.10.b Software Engineering: Design notations and documentation; D.3.2.k Programming Languages: Macro and assembly languages; D.4.6 Operating Systems: Security and Privacy Protection

Author Keywords
Assembly Language, Collaborative Documentation, Shared Comprehension Artifacts

INTRODUCTION
Reverse engineering is complex and time-consuming, particularly when obfuscated in malware. The current lack of assembly language tool support further exacerbates this problem. Surveys and interviews show the apparent need for better means of documentation within this domain [16]. This paper investigates this issue, and establishes if this documentation could be collaborative in nature.

It is our hope that providing support for collaborative documentation will help reduce the cognitive overload associated with malware analysis. This is by reducing the amount of work each analyst will need to perform through information sharing, as well as by providing better documentation support within the tools. Such support would include those that would aid automated report creation.

The contributions of this paper, in regards to malware comprehension, are:
1. a survey evaluating collaboration and documentation workflow, and associated tools
2. a prototype tool to evaluate the use of comments over shared comprehension artifacts
3. an initial investigation of the proposed approach through interviews

This paper is organized as follows. Section 2 discusses related work. Section 3 presents the survey results and Section 4 focuses on the prototype that we have created in response to the survey. Section 5 provides the initial feedback obtained through interviews. The paper closes with future work and a conclusion.

RELATED WORK
In previous studies [15, 16], we investigated the issues faced by malware analysts and found that documentation (which included notes, bookmarks and tagging) was mentioned by 2 out of 15 (13%) of the respondents, but was mentioned far more often in interviews, which led us to look further into what support is currently available. There are two tools, that we are aware of, that try to address collaboration within reverse engineering tools. These are BinCrowd by Zynamics [1] and CollabREate [18, 13].

BinCrowd provides a database that can be used to share disassembly information with team members. This information can then be shared via a web interface or through an IDA Pro plugin [19]. It works by allowing an analyst to document their code and then upload these comments to the database. BinCrowd then uses its BinDiff format to find related functions, and files, as well as comparing files and displaying statistics about them. These comments are inline within the code using IDA Pro’s commenting capability which only allows comments around functions. Though reasonable as a means of documentation for a single developer, this form of commenting may not be conducive to discussion and collaborative documentation.

Figure 1 shows the importing of comments to functions within IDA Pro using the BinCrowd plugin. We can see the box that
shows retrieved function information, in this case, two functions that are identified as having a high match quality.

CollabREate works somewhat similarly to BinCrowd in that it uses an IDA Pro plugin and pushes changes to the IDA Pro database files (IDB) to a server. Any user that is subscribed to the same CollabREate project will receive these updates, and therefore remain synchronized. These IDB updates include comment changes, but also quite a few more including adding or deleting functions, enums and structs. Again, these comments surround functions, but may not be sufficient to support collaboration.

As for tools that allow comments over shared artifacts, since this is a new area, it is hard to find related work, particularly in low level systems support. One example is Google’s SideWiki [3] which is a browser sidebar that lets you contribute and read information alongside any web page. This is shown in Figure 2, where the comment thread is shown on the left pane. This figure shows a comment thread regarding Google’s home page. Users can add information such as background, tips and perspectives to annotate the web content.

USER SURVEY
We invited security analysts by email, to take a small survey to find out how documentation and collaboration are currently used and whether or not the tools in use are sufficient. The emails were sent to people we had contact with that were able to freely forward it to others they thought would be interested. We had 6 respondents, both from the reverse engineers in industry and academia. The survey was issued via Google Forms [12], contained 11 questions in total, and was expected to take 10 minutes to complete. The results are discussed below for each topic, and a summary and analysis are provided at the end of this section.

Collaboration
This part of the survey contained 6 questions that first asked whether or not there was a need to collaborate, how the collaboration occurred and what tools were currently used for that collaboration. Finally, we asked whether or not these tools were sufficient and what features they would like to see. The results for each question are discussed in this section.

When you’re working at the assembly level, do you have a need to collaborate with others? This question was given as a simple yes or no answer. 4 out of 6 respondents (67%) said that yes, they do collaborate while 2 out of 6 respondents (33%) do not.

If yes, how do you collaborate? Of the 4 out of 6 respondents who did collaborate, they did so by talking to each other (including giving each other crash courses on what they had done) and sharing information. In one case, this was achieved by working on the same computer.

For document sharing, 3 out of 4 (75%) mentioned that they shared IDA Pro [19] database files (IDB files) which include documented pieces of code, whereas 2 out of 4 (50%) mentioned sharing information from other sources as well, such as documents, internet links, and reports.

If yes, who do you collaborate with? Do you collaborate with them all at once? Of the 4 out of 6 respondents who did collaborate, all of them did so with coworkers (members of the same team) or co-researchers. All of them collaborated in small groups, with one or two colleagues at a time and 3 out of 4 (75%) also collaborated all at once in a large group. One respondent mentioned that the large group would be to share end results, so it may be that the group size is dependent on the collaboration intent.

If yes, what tools do you currently use to support your collaboration? Of the 4 respondents, 3 out of 4 (75%) did not mention any software tools, though all respondents did use...
devices such as phones, presentations on projectors, email for sharing documents as well as memory sticks and shared folders. 1 out of 4 (25%) had tried CollabREate, but the tool was not usable in practice since their malware analysis required them to be on isolated computers not connected to the internet for security purposes.

Are these tools sufficient? Again, this was a yes or no answer. 5 respondents answered this particular question. 4 out of 5 (80%) said that no, the tools are not sufficient while 1 out of 5 (20%) said that they were.

If not, what features do you wish you had for collaboration in your tool environment(s)? Or if you don’t use any tools, what features do you wish you had? Answers to this question varied but answers included:

- synchronize documentation efforts by importing pieces that have already been documented
- share comments on specific parts of code
- share renamed variables and functions
- retrace steps that an analyst did that were useful
- status updates on what an analyst just did or found, such as “X just renamed sub_0000ABCD to XYZ”

Documentation
This part of the survey contained 5 questions, asking whether or not there was a need to create documentation and if so, what kind of documentation and what tools were used to create it. We then asked whether or not these tools were sufficient and what features they would like to see. The results for each questions are discussed below.

When you’re working at the assembly level, do you have a need to create documentation? This was a simple yes or no answer, and 6 out of 6 respondents (100%) said that yes, they do create documentation.

If yes, what form does the documentation take? If you can, please post a sample in the box below. Otherwise please comment on what a sample of documentation would look like. The most prevalent answer to this question was comments within IDA Pro which 4 out of 6 respondents (67%) mentioned. These comments are inline within the assembly code but can only surround functions. 1 respondent mentioned that they rely on making the comments appear semi-Javadoc [6] so that they are easier to parse. 1 respondent also mentioned that comments are made on paper.

4 respondents (67%) mentioned analysis summaries within reports while 2 respondents (33%) mentioned that they create these reports with code snippets of the disassembly. 1 respondent also added that a sequence diagram might be included in the report to show the call sequence leading to the problem.

If yes, what tools do you currently use to support your documentation? While 2 of the 6 respondents (33%) reported using no tools at all, an equal number (33%) used Microsoft Word [14], Excel[8] and OneNote [9]. One use of Excel was to create control flow graph representations using hyperlinks.

Tools mentioned by 1 respondent each included LaTeX [7], Visio [10], OneNote, IDA Pro, and a wiki.

Are these tools sufficient? Given as a yes or no option, 5 out of 6 respondents (83%) said that no, the tools were not sufficient while 1 out of 6 (17%) said that they were.

If not, what features do you wish you had for documentation in your tool environment(s)? Or if you don’t use any tools, what features do you wish you had? There were 4 responses to this question. 2 of the 4 (50%) mentioned being able to integrate information from various sources, including from reverse engineering tools. This would include pulling information from different sources (i.e. IDB files, API documents, diagramming tools) to consolidate into one report. 1 respondent mentioned that this may be possible through tagging.

2 respondents (50%) also discussed the need for comment support. One discussed tagging and grouping of comments as well as searching for them. This was mentioned in particular because IDA Pro only allows comments at the beginning and end of functions. The tagging would allow comments on specific code, basic blocks and variables. The other respondent mentioned having to annotate a snapshot of the graph view in IDA Pro, which is also likely due to IDA Pro only allowing comments surrounding functions. This graph view is shown in Figure 3. This respondent also mentioned the need to make integration automated by being able to automatically export comments to the documentation system so that it remains up-to-date when changes are made in IDA Pro.

1 of the 4 respondents (25%) also mentioned being better
Collaboration

67% (4 out of 6) is needed
80% (4 out of 5) current support is insufficient

Features requested:
- Share existing documentation
- Retrace others’ steps
- Status updates on analysts’ actions

Documentation

100% (6 out of 6) is needed
83% (5 out of 6) current support is insufficient

Features requested:
- Better comment support
- Better documentation integration
- Document execution paths

| Table 1. Summary of User Survey.

able to document execution paths, as well as creating small videos to explain concepts that will need to be repeated, such as how we reach a location and the path that led there and why a function is being renamed.

Survey Overview and Analysis

Table 1 shows a summary of the results from the user survey. Collaboration was reported as necessary by 67% of respondents, however 80% of those that replied reported that their tools were not sufficient. The features requested included being able to share existing analysis documentation, including comments and renamed variables and functions, retrace steps that another analyst took as well as follow status updates on analysts’ work.

As for documentation, 100% of respondents needed to create documentation while 83% reported that the tools were insufficient. Respondents wanted better support for comments within the reverse engineering tools, as well as better integration overall between the different tools that they used for documentation. This included the need to document execution paths.

In the following section, we discuss the prototype we have developed that takes some of these issues into account.

PROTOTYPE: DESIGN AND IMPLEMENTATION

In order to address some of the issues discovered in the survey, namely the lack of commenting support, including importing existing documentation and documentation of execution paths, we built a comments view into a prototype tool we call Tracks. Tracks was designed to show static as well as dynamic control flow for assembly language in the form of sequence diagrams. The functionality and implementation of Tracks is discussed in depth in [16]. A sample Tracks diagram is shown in Figure 4, which shows lifelines as the white boxes with green lines vertically underneath and calls as the arrows between them. This shows a small part of a larger sequence diagram for a static path through calc.exe, a simple calculator example. At the top of this figure, a panel shows which file a function is defined in. Here we can see that LocalFree comes from KERNEL32.dll, and it is denoted with a purple I icon. The small thumbnail view at the bottom is used to quickly navigate around the diagram, useful for extremely large traces. The tree view on the left shows all of the functions defined within calc.exe.

In order to add collaborative documentation support, we added the ability to create comments on calls, lifelines and cycles within the diagram through double-clicking. Cycles are repeated patterns of calls and are shown as shaded boxes containing calls. A preference is given to the user to import the comment data when the sequence diagram files are loaded. If comments exist, the artifacts are marked with color-coded stars. Red stars indicate that there is little activity and is used for items with less than 10 comments. Yellow stars indicate a medium level of activity and are used when there are more than 10 comments but less than 25. Green stars are used to indicate high activity and are used when there are 25 or more comments. This color selection is based on the download health colors commonly used in bit torrent applications, where green represents that the file is available from many other users to download from.

The modified diagram is shown in Figure 5. In this screenshot, we are investigating the Mariposa botnet [23, 21] and the trace file for server communication. The left panel shows the Tracks sequence diagram with comments on the two lifelines to the far right, indicated by the red stars within the lifeline boxes. The figure also displays a cycle, which is outlined in red and filled in with grey, and contains the text “1 comment” to the far left of it. The right panel contains the comments view, which shows the ID of the thread at the top, the logged in user below that as well as the box to enter comments, followed by the comment thread itself.

Tracks was built using the Eclipse framework [22] and an open source, extensible tool called Diver [17, 20]. It also
communicates with IDA Pro via an IDA Pro plugin. This plugin enables IDA Pro and Tracks to pass messages back and forth, which include static control flow information, events for navigation and tracing, and setting preferences.

The comment functionality was built as a web application using Google App Engine [2]. Comments are stored in the database with an ID of the artifact it pertains to. We then use this ID to build the comment threads through a query, and retrieve the comment count. We also use the google account login to add user information to each comment. This web application was then incorporated into Eclipse as a view that contains a browser.

INITIAL FEEDBACK: EXPLORATORY INVESTIGATION

In order to gain first impressions of the proposed approach, we contacted 5 survey participants that gave us permission to do so. Of those 5, we conducted telephone interviews with 3 of them who responded for approximately 30 minutes each. They were issued a demo video in advance¹ that showed the commenting features. Participants were able to freely respond with comments but some of the questions we asked included:

- Is there anything particularly useful in the demo?
- Is there anything not useful?
- Are there any features missing?
- Do you think the stars/colors are the best representation?
- Are there other artifacts you would like to be able to add comments to?
- Do you think there is a need to create documentation, for personal use, in the same fashion?
- How many comments do you foresee being in a thread?

The 3 interviews are summarized in the following subsections, followed by a brief overview and analysis of these results.

Interview 1: Comments on Basic Blocks

Rob² typically does not work on projects with other people but his clients do. He believes that reverse engineering tools fall down without collaboration but also remarks that this is changing as more tools are adding support.

As for usefulness, he finds that the history of comments is useful and that it is similar to having a wiki alongside the diagram. He also likes the ability to see comments by their post date, but would want to be able to navigate from the newest comment to where it pertains to in the code.

¹The demo video is available at http://jenniferbaldwin.info/ava/CommentDemo.mp4

²The names used in this study are fictitious.
Extra features he would like to see include being able to incorporate comment threads directly into a report either by exporting it or printing it to PDF. He often puts snippets of code into reports alongside the documentation, which he would like to have happen automatically. Rob would also want to be able to look at the differences in comments between two executables, for example if a new version of an executable is released. He would also like to be able to link comment threads, for example if code is duplicated in another executable, and to have the comments themselves link to the internet or to files. Finally, Rob could see the potential to have comments approved, for example by a team leader. This means that if a particular part of the code has been approved by reaching a level of saturation of comments, then no more analysis would need to occur on it. In this way it becomes a tool for managing workflow.

As for other artifacts, he often comments basic blocks of code, for example, if statements and would like to be able to add comments to these. One idea for this would be to add boxes to the lifeline image that could be double-clicked to load their comment threads.

Rob mentioned that having the ability to create private documentation (documentation for personal use) would be useful as a personal workspace for rough notes.

As for the length of comment threads, this was difficult to answer but Rob suggested that having a tree hierarchy or nested overlay for long comment threads and thought that interesting sections might have 8 to 12+ comments.

Interview 2: Comments on Mouseover

Joe usually works as an individual, with maybe one other person at a time. However, he does not work in a group so he is not used to working in this context and has a greater need to create documentation privately. He usually creates documentation with the comments in IDA Pro, but does find the ability to ask questions to other people interesting.

Joe believes that double clicking to have the comments load on the right is disruptive to the workflow and instead suggests that they appear on mouseover with a summary of comments in place. In this way, the user does not have to navigate to the side panel and back again, wondering what they were looking at, especially in a large trace.

He would also like to be able to pull in information from other sources. For example, he is almost always looking up functions on MSDN [11] since he forgets what the parameters are. The first thing he would like to do if he had the prototype code is to add links to it from the comments. Upon further investigation, there is an IDA Pro plugin that provides this functionality [5]. A screenshot of this tool having...
imported functions is shown in Figure 6. Here we can see that MSDN documentation describing function parameters has been pulled in as comments into the code.

When asked about the use of the color-coded stars, he did mention that numbers might be more useful and that highlighting the most recent would be helpful, with perhaps an exclamation mark. He also thinks that people will adapt to whichever colors or methods are used.

**Interview 3: Tracks as an Authoring Tool**
Mark finds the comments view pretty interesting and believes it would be useful with even only 2 to 3 people in a team. However, he would need to have a local server to support the comment threads since the machines used for analysis are standalone (not connected to the internet) due to security reasons. He would also need to be able to link to the comments in IDA Pro, otherwise there would be 2 sets of comments spread out over 2 tools.

He does not think that the colors would be useful as they probably would not reach 5, 10 or 25 comments within their team. Mark mentions that a metric that might be better is the level of certainty of a comment. For example, red might be asking a question while green is a response to a question.

As for the granularity of commenting artifacts, he believes that for sequence diagrams, commenting calls, lifelines and cycles are pretty good. However when he makes comments in IDA Pro, he often comments particular instructions.

Private documentation was seen as useful and especially so if it can be saved locally and shared offline.

An interesting use of the Tracks tool that Mark mentions is the ability to add and remove calls from both the static diagram and the dynamic traces. In this way, Tracks can be used as a sequence diagram authoring tool. Currently they are using Rational Rose [4], but he is not satisfied with it, and would also like to have integration with IDA Pro.

**Interview Overview and Analysis**
Table 2 shows a summary of the results from the interviews. We split up the results by question, but major points include the ability to have comments stored on a local server, being able to link comment threads with one another and with existing IDA Pro comments, being able to manually edit the sequence diagrams, and flexibility of what elements can have comments added to them (i.e. basic blocks).

**FUTURE WORK**
There are many avenues for future work but some of the main categories include better integration with other reverse engineering tools, changes to which artifacts can be commented, and changes to the flags that are used to alert the user to the presence of comments.

For integration, since most users currently create their comments inline in the assembly code within IDA Pro, we would need to be able to export those into the comment threads.

It is important that we can also import documentation from other sources, such as from MSDN. Another important integration technique will be the use of sequence diagrams within reports. This means that users will need the ability to manually edit sequence diagrams that they can then save to an image and import into their documentation. The ability to export a sequence diagram as an image is already built into the Diver framework so it simply needs to be enabled in Tracks. Additionally, the ability to link a code snippet to the comment thread so that it can be exported to documentation may be of importance.

As for the granularity of what can be commented, we need to be able to support some level of arbitrary comment placement. For example, users need to comment parts of the code such as basic blocks and particular instructions. We need to indicate placement of such comments within the sequence diagram. To alert the users to the presence of comments, we may look at metrics other than the sheer number. Other metrics that presented themselves were the most recent posts as well as the certainty of the post (e.g. question versus answer).

We will also need to consider the scalability of comment threads and the amount by which the threads will grow in practice. If size becomes an issue, we may need to look at other ways to represent them, such as with a tree hierarchy or with nested overlays. We may also need to have the comments window show up on top of the sequence diagram if we do not want to interrupt the work flow in a large trace.

| **Is there anything particularly useful in the demo?** |
| History of comments |
| Post date to sort by newest comment |
| Ability to ask questions (have a discussion) |
| Useful even with a small team |

| **Is there anything not useful?** |
| Comments on mouseover |
| Would need a local server for standalone machines |

| **Are there any features missing?** |
| Navigate from comment to code |
| Automatically extract code/comments to report |
| Differences in comments between two executables |
| Link comment threads |
| Comment approval |
| Link with existing IDA Pro comments |
| Manually edit sequence diagram |

| **Do you think the stars/colors are the best representation?** |
| Number of comments |
| Highlighting most recent |
| Certainty of comment |

| **Are there other artifacts you would like to be able to add comments to?** |
| Basic blocks |
| Instructions |

| **Do you think there is a need for private documentation in the same fashion?** |
| 100% (3 out of 3) Yes |
| Good place for rough notes |
| Save locally and share offline |

| **How many comments do you foresee being in a thread?** |
| Tree hierarchy or nested overlay |

**Table 2. Summary of Interviews**
As for the comment capability itself, the ability to categorize, tag and search for posts will need to be added. The tagging can lend itself to inclusion within other documents but also to link comment threads and be used for searching. We also need to allow users to link to external information such as internet links, and files.

We also need to provide support for private documentation that can be saved locally and shared offline. This gives the user a private area in which to save notes. This might be implemented as a separate tab within the comments view in much the same way as the collaborative approach.

Other areas of future work include updates on what other analysts are doing, support for comments on a private server, retracing steps that an analyst took, the difference listing between comments on two particular executables and comment approval.

CONCLUSIONS
Our research goals include developing tools and user interfaces that will assist reverse engineers working in cyber security. We have shown that both collaboration and documentation are important factors in this domain. We have also shown that both areas are insufficiently supported by current software tools. By creating a prototype that adds comment threads to artifacts in an existing sequence diagram tool, we were able to address some of the concerns brought up by developers in our survey. Further, the use of this prototype serves as a means for discussion for further development of support for collaborative documentation.

REFERENCES