Next Generation Collaborative Reversing with Ida Pro and CollabREate

Christopher Eagle and Timothy Vidas

Department of Computer Science,
Naval Postgraduate School, Monterey, CA
{cseagle,tmvidas}@nps.edu

Abstract

A major drawback with the use of most reverse engineering tools is that they were not designed with collaboration in mind. Numerous kludgy solutions exist from asynchronous use of the same data files to working on multiple copies of data files all of which quickly diverge, leaving the differences to somehow be reconciled. These methods and existing tools[1] provided a first step towards automated collaboration amongst IDA Pro[2] users, however they suffer from several shortcomings including the fact that tools have failed to keep pace with the evolution of IDA's internal architecture. In this paper the authors present a new collaborative tool, titled collabREate[3], designed to bring nearly effortless collaboration to IDA users.

1 Introduction

Reverse engineering of binary programs is a very time consuming task, often performed in a very serialized manner. Analysts are forced to operate in this serial fashion in large part because of the nature of the tools they choose to use. When several analysts do attempt to coordinate their efforts, it is usually accomplished by sharing copies of files or through strict turn taking access to a single shared work file. In some cases (predominantly with text files), work files may lend themselves to storage within a versioning system such as CVS [4] or subversion [5]. Due to the binary nature of most reverse engineering situations, it is no easy task to provide any automated merging of changes to such files.

1.1 IDA Pro

One of the premier analysis tools for analyzing binary executable files is the Interactive Disassembler Pro (IDA Pro, or simply IDA) from Hex-Rays [2]. IDA Pro facilitates disassembly of program binaries by a single user and stores its work in data files which utilize a proprietary, binary format that does not easily lend itself to incorporation into a revision control system. In this regard, the format of IDA data files (IDB files) is analogous to compressed archive formats such as ZIP and TGZ.

When two or more analysts wish to coordinate their efforts in analyzing a single binary file, they must either collaborate at a single workstation, take turns working on a single, shared IDB file, or painstakingly merge updates from several disparate IDB files into a single unified file that contains information from all related working copies. Needless to say, such processes may be both slow and error prone.

Complicating matters is the fact that Hex-Rays tends to abandon backward compatibility when releasing new versions of IDA. For example, not only can IDB files created with the current version of IDA (version 5.2) not be opened with older versions of IDA, but the current version does not provide the capability to save an IDB in a way that older versions of IDA can open the file. This makes it extremely difficult for analysts using different versions of IDA to coordinate their efforts using anything other than face to face interaction and manual change incorporation. Clearly this is not an ideal situation for analysts who reside in different geographic locations.

1.2 IDA Sync

In March 2005, in an effort to address the need for a collaborative reverse engineering tool, and Pedram Amini released Ida Sync [1]. The goal of Ida Sync was to provide a means for IDA users to share their work by posting certain IDB updates to a central project server which would store and forward each update to additional IDA users subscribed to the same project. Because updates are stored in a format independent of the IDA version used to generate the updates, Ida Sync made it possible for users with different versions of IDA

to coordinate their activities to the extent allowed by Ida Sync.

Ida Sync operates as a plugin module to IDA Pro itself. The Ida Sync architecture introduced new Ida Sync specific command sequences (hotkeys) that mirror several common IDA operations including insertion of comments into IDB files and changing the names assigned to program virtual addresses and local variables. This approach has several drawbacks including the fact that it requires Ida Sync specific changes to a specific IDA Pro configuration file and that it requires IDA users to learn, and remember to use, additional command sequences in order to share changes to their IDB files. Another drawback is that users are limited to sharing only the operation types for which Ida Sync has chosen to implement command sequences. Additional changes such as reformatting code to data, data to code, or changing the format of an instruction's operands must be shared manually or not at all. A final drawback is that Ida Sync fails to account for any changes made to an IDB file through the use of IDA's scripting or plugin interfaces, thus a user who makes use of a script to modify an IDB file, must share that script with every other user working on the same project and expect those users to execute the script at the appropriate time in order to remain in sync with the project.

2 The IDA SDK

As an IDA plugin, Ida Sync relies upon the Hex-Rays software development kit (SDK) for IDA in order to capture user actions and forward them, behind the scenes to a central Ida Sync update server. In the case of Ida Sync, the SDK facilitates this process by providing an API for interacting with IDA which in turn activates the Ida Sync plugin each time a user activates an Ida Sync specific hotkey sequence. Ida Sync, in turn, determines which hotkey sequence the user has activated, then performs actions that mimic, in some sense, a standard IDA action.

For example, when a user activates the Ida Sync comment insertion command, Ida Sync displays a dialog very similar to the standard IDA comment insertion dialog. This familiarity helps the user draw on their experience with IDA to complete the requested operation. When the user signifies that the operation is complete, Ida Sync creates a comment in the user's IDB file, just as IDA would, and takes the additional step of forwarding the comment's location and content to the Ida Sync server.

Ida Sync updates received by the Ida Sync server are forwarded to other connected Ida Sync clients, where the Ida Sync plugin running on each client interacts with the IDA API to apply the update to the local IDB file with no user interaction required.

The IDA API (as published via the SDK) offers a more natural means for intercepting user actions than the introduction of additional hotkey sequences as implemented in Ida Sync. The API provides an event registration and notification mechanism that allows plugin components to register interest in specific types of IDA events and receive notification when those events take place. In the context of collaborative reverse engineering, such notifications allow a plugin programmer to seamlessly hook into a user's actions and forward information concerning each action to other interested users.

Since the introduction of Ida Sync on 2005, the API's capability in regards to event notifications has grown significantly. Specifically, with the introduction of IDA version 5.1, the number of available notifications was expanded significantly to include notifications for a large percentage of user initiated actions. Through the use of the expanded notifications API, it has become possible to silently observe the majority of user modifications to an IDB file and forward detailed information concerning each change to any interested user.

Through the use of such an approach, it becomes possible for users to synchronize their changes without requiring users to edit IDA configuration files or requiring them to learn (and remember to use) any new commands. As an added advantage, notifications are also triggered for actions initiated via the execution of IDA scripts or other plugin modules, eliminating the need to share and synchronize the use of these resources.

3 CollabREate

The collabREate plugin represents the next generation in IDA collaboration. The goals of the collabREate project are to provide a deeper and more natural integration of the plugin component within the IDA API and to provide a more robust server component backed by a SQL database and capable of supporting features beyond simple database synchronization.

From a high level perspective, collabREate owes much to the Ida Sync project. The collabREate plugin processes databases updates and communicates with a remote server component to synchronize database updates with additional project members. The

asynchronous communications component functions in a manner similar to that used by IDARub[6] and Ida Sync, however it has been improved considerably in order to perform properly under heavy load. This is where the similarities to Ida Sync end.

3.1 CollabREate Architecture

The collabREate plugin takes a fundamentally different approach to capturing user actions by leveraging IDA's event notification mechanisms. Rather than introducing replacement commands that must be configured and remembered by users, collabREate hooks various database change notifications and seamlessly propagates database updates to the collabREate server. The types of database updates that can be captured and published by IDA are dependent on the IDA version which happens to be running the plugin. A summary of actions supported by recent IDA versions can be found in Table 1.

The collabREate architecture offers publish and subscribe capabilities to participating users. A user may selectively choose to publish their changes to the collabREate server, or subscribe to changes that others post to the server or both publish and subscribe. For example an experienced user may wish to share (publish) their changes with a group while blocking (not subscribing to) all changes made by other users. These capabilities can be specified at the user level, the project level, and ultimately individual session level allowing for very granular and flexible control. For example, a novice user may wish only to publish comments, while another user may wish to subscribe only to name changes and patched byte notifications.

Another major feature of collabREate is the ability to circumvent IDA database incompatibility issues. Databases produced using IDA version 5.2 can not be opened using older versions of IDA. This incompatibility eliminates one of the primary forms of IDA collaboration, database sharing, in cases where some users do not own the newest version of IDA. However, by synchronizing changes through a collabREate server, users of older versions of IDA are able to receive updates generated by users with newer versions of IDA. Unfortunately, users of older versions of IDA are also least able to publish their changes, so the flow of information is somewhat one way.

3.2 CollabREate IDA Pro Plugin

One of the most significant features of the collabREate plugin is its degree of integration with the IDA SDK. IDA notifications are tied to specific database actions, not specific user actions. The fact that user actions happen to trigger IDA notifications is of course critical to the collaborative process, however notifications can be triggered by other means. IDC scripts and API function calls can generate notification messages as well. As a result, the actions of an IDC script that patches database bytes, renames locations or variables, or inserts new comments will be published to the collabREate server and will ultimately be shared with other IDA users working on the same project.

IDA Version	**4.9(fw) 5.0**		**5.1**		**5.2**	
Action (Publish/Subscribe)	**P**	**S**	**P**	**S**	**P**	**S**
Undefine	✓	✓	✓	✓	✓	✓
Make code	✓	✓	✓	✓	✓	✓
Make data	✓	✓	✓	✓	✓	✓
Move seg	✓	✓	✓	✓	✓	✓
Name changed		✓	✓	✓	✓	✓
Func added or deleted		✓	✓	✓	✓	✓
Func bounds changed		✓	✓	✓	✓	✓
Byte patched		✓	✓	✓	✓	✓
Comment changed		✓	✓	✓	✓	✓
Operand type changed		✓	✓	✓	✓	✓
Enum created or changed		✓	✓	✓	✓	✓
Struct created, deleted, or changed		✓	✓[1]	✓	✓[1]	✓
Func tail added or deleted		✓	✓	✓	✓	✓
Seg added, deleted, or changed		✓	✓	✓	✓	✓
Flirt function identified		✓		✓	✓	✓

Table 1: collabREate capabilities by IDA version

The collabREate plugin stores some information in the IDB itself, particularly information that ties the IDB to a particular collabREate project and information about the most recent communication with the collabREate server. This allows for a more intuitive interface when reconnecting to an existing project. If a user reconnects to a project they have previously

1 In order for full structure updates to be properly published IDA 5.2 and an updated IDA 5.2 kernel is required (available upon request to Hex-Rays.)

worked on using an IDB that already contains project information, the server forwards all updates that have not yet been applied to that IDB. Since the state of the IDB is tracked in the IDB (instead of on the server) the same user account can have multiple collabREate sessions open, or start a new collabREate session without the worry that some updates will be missed because the severer believes that the user account has already received the updates. Additionally, should a user wish to abandon the changes they have made to a database by closing the database without saving their work, there will be no confusion on the server side regarding which updates the user may or may not have received the next time the database is opened.

Table 1 highlights the fact that older versions of IDA are not capable of publishing as much information as more recent versions of IDA. This is a result of the evolution of IDA's SDK over time. Specifically, a significant number of new notification types were added beginning with version 5.1 of the SDK. An important feature of collabREate is the fact that the inability to send all forms of updates does not prevent a version of IDA from receiving all forms of updates.

A specific problem that needed to be addressed in the plugin was the fact that the act of applying a received collabREate update to a database would cause IDA to generate a new notification message in the database to which the update was applied. In order to prevent duplicate update message from being sent to the server and all associated users, the collabREate plugin temporarily disables reception of IDA notifications whenever a received update is being applied.

3.3 CollabREate Server

The collabREate server offers two modes of operation. A basic mode and a database mode. When the server is unable, either intentionally or unintentionally, to connect to a backend database, the server enters into its basic mode of operation. In basic mode, the server functions as a simple non-authenticating, update reflector. Each time an update arrives for a given project, the update is forwarded to all other users that happen to be connected to the same project. In basic mode, no provisions exist for persistent storage of individual updates. Users who are late to join a basic mode project will not receive any updates that have been made prior to their arrival.

In database mode, the server makes use of the persistence features of a backend SQL database to provide user management and authentication as well as allowing for multiple projects to be associated with a given input file, and persistent storage of all updates made to each project.

The collabREate server component is currently implemented in Java and utilizes JDBC [7] to communicate with a back end SQL database. The server is responsible for user and project management. User accounts are managed via a separate management interface to the server, while projects are created by users as they connect to the server. The separated server management component allows the server to be run as daemon.

The server contains as much of the decision logic and state tracking as possible. This both reduces the chances that a rogue plugin is able to perform actions that it shouldn't and it leaves the plugin to focus on the core ability of generating messages when events occur and applying updates as they are received from the server.

The collabREate server has the capability of *forking* existing projects to allow users to create alternate branches of a project without impacting other users. This is a useful feature if you wish to make (and track) a significant number of changes to a database without forcing those changes on other users. In the collaborative spirit, users currently collabREating on a project are given the opportunity to follow a user that decides to fork, or they can choose to remain in the parent project. As the server is capable of handling multiple projects related to a single binary input file, the plugin and the server take additional steps to ensure that users are connecting to the proper project for their particular database.

It is well known, that IDA provides no “undo” capability [8]. Similar to a project fork, the collabREate server allows for a feature called a *checkpoint* (aka *snapshot*). A checkpoint allows a user to apply a name to the particular state of the IDB present in their instance of IDA. A user may choose to create a checkpoint prior to performing some complicated or questionable edits to the IDB. If these updates result in an unwanted IDB state, users can choose to abandon that particular project by closing that IDB, opening the original binary, and forking a new project using the checkpoint as a starting point at which point the server will send the user all updates up to the checkpoint.

A final feature of the collabREate server is the ability to restrict users to specific types of updates. For example, one user may be restricted to a subscribe only profile, while another user may be allowed to publish only comments, while a third is allowed to publish all types of updates.

3.4 CollabREate Protocol

Communication between the plugins and the server occurs asynchronously. The collabREate protocol is a simple binary network protocol. The basic form of a datagram is shown in Table 2. The UpdateData field varies in structure, for example there are several commands that don't effect IDA at all, but are used for plugin to server communications such as user login and project selection. These collabREate specific datagrams do not contain the UpdateID field.

Field Name	Size	Description
Size	4 bytes	Size of this datagram (inclusive)
Command	4 bytes	Indicates type of communication
UpdateID	8 bytes	Unique ID assigned per update
UpdateData	Varies	The actual update Data.

Table 2: Basic collabREate datagram structure.

When the server is operating in database mode, all datagrams relating to IDA database modifications are stored into the SQL database. This allows for their eventual retransmission to users that connect to the associated project and request any updates that they may have missed.

3.5 Example CollabREate Session

Depending of a variety of factors, the process of setting up a collabREate session varies. As an example, a typical sequence of events is provided.

First, a collabREate server is is launched. For this example, we assume the server is operating in database mode. Next, the plugin is activated by an IDA user which prompts the user to enter server host information. The user authenticates with the server using standard techniques [9][10]. Following authentication, the collabREate plugin sends the MD5 hash of the input file that the user is analyzing to the server. The MD5 value is used to ensure that multiple users are in fact working on identical input files. Several projects might share the same MD5 (indicating that there are several projects relating to the same original binary for whatever reason). The server sends back a list of projects and checkpoints to the plugin. At this point, the user can choose to join an existing project, create a new project, or fork a new project from a checkpoint. For any of these methods, the user may indicate the types of updates to which they would like the plugin to publish and subscribe. At this point the IDB is tied to the project using a unique identifier to facilitate reconnecting to the project at a later time.

Once a collabREate session is established, users can largely work within IDA as they normally would. Updates are broadcast to all other users connected to same project as they occur.

Attempting to activate the plugin a second time (via Hotkey or menu item) results in a modal dialog box presenting collabREate specific commands that allow the user to fork, create a checkpoint, manage permissions, disconnect, etc.

4 Future Work

A number of features remain on the collabREate "todo" list including but not limited to the following:

1. Web interface for administration of a collabREate server including the ability to add/remove/edit users, and well as delete or archive projects.
2. More granular client and server side permissions.
3. Provide a means for disconnected users to cache updates for later merging once a connection to a server is re-established.
4. Project migration/replication across different collabREate servers.
5. Add revert capability to the checkpoints.
6. Provide an XML export feature for update content.

5 Conclusions

CollabREate is a step in the right direction for reverse engineers requiring a means to share their work among several users, across several locations, or across multiple versions of IDA. Leveraging the capabilities of the IDA SDK allows collabREate to provide a very low learning curve for new users without compromising the degree of supported IDA features.

In the future, it is anticipated that the evolving nature of the IDA SDK will facilitate additional useful features for both groups and individuals such as an undo-like capability.

References

[1] Ida Sync. P. Amini. `http://pedram.redhive.com/code/ida_plugins/ida_sync/`

[2] IDA Pro.
http://hex-rays.com/

[3] CollabREate. C. Eagle, T. Vidas.
http://www.idabook.com/collabreate

[4] CVS – Open Source Version Control.
http://www.nongnu.org/cvs/

[5] Subversion version control system.
http://subversion.tigris.org

[6] IDARub. Spoonm.
http://www.metasploit.com/users/spoonm/idarub/

[7] The Java Database Connectivity API.
http://java.sun.com/javase/technologies/database/

[8] Eagle, Chris. The IDA Pro Book. San Francisco: No Starch Press, 2008.

[9] CHAP. RFC 1994

[10] HMAC. RFC 2104