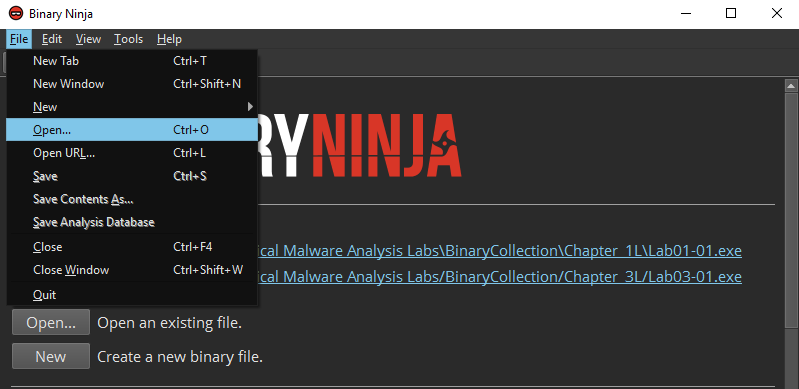
Binary Ninja Tutorial

Loading an executable:

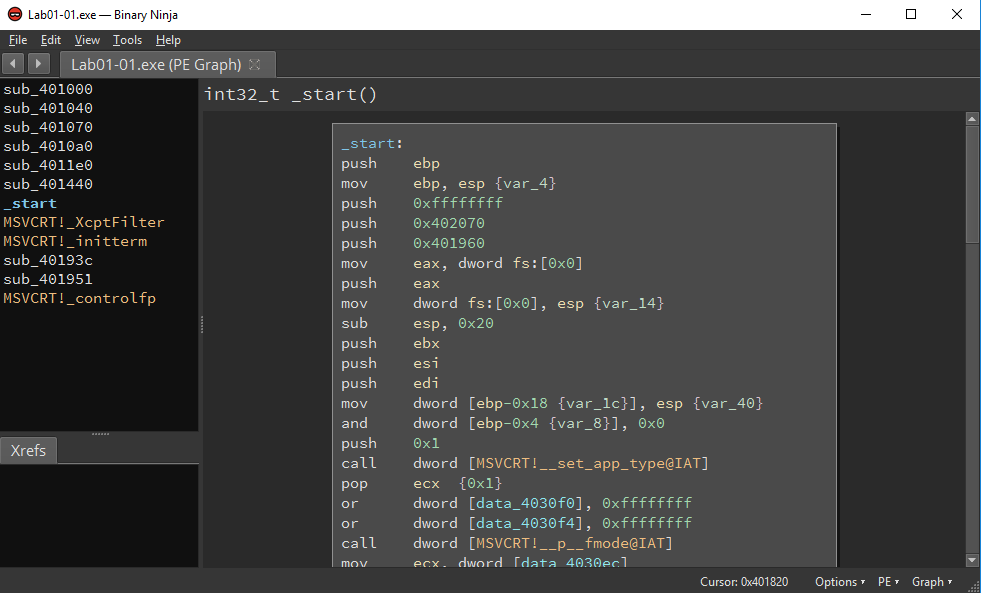
Binary Ninja offers several methods for loading an executable into the program. The easiest method involves simply dragging the file that you want to analyze onto the Binary Ninja icon. This will automatically launch the application and bring up the default view for analysis. An alternate method for loading a file is using the File -> Open option as shown in figure 1. From here, the file explorer will open. Find the appropriate file and press open. This is functionally identical to the first method, so either will work. Note that some larger samples will take time to analyze but the interface will remain functional.



**Figure 1- Opening a file**

The Binary Ninja Interface

Binary Ninja offers a wide range of features in a more user-friendly interface than some other disassembly programs. With a focus on a clean user experience, some options may appear hidden until further inspection. Figure 2 shows the initial screen that a user will be presented with upon opening a malware specimen.



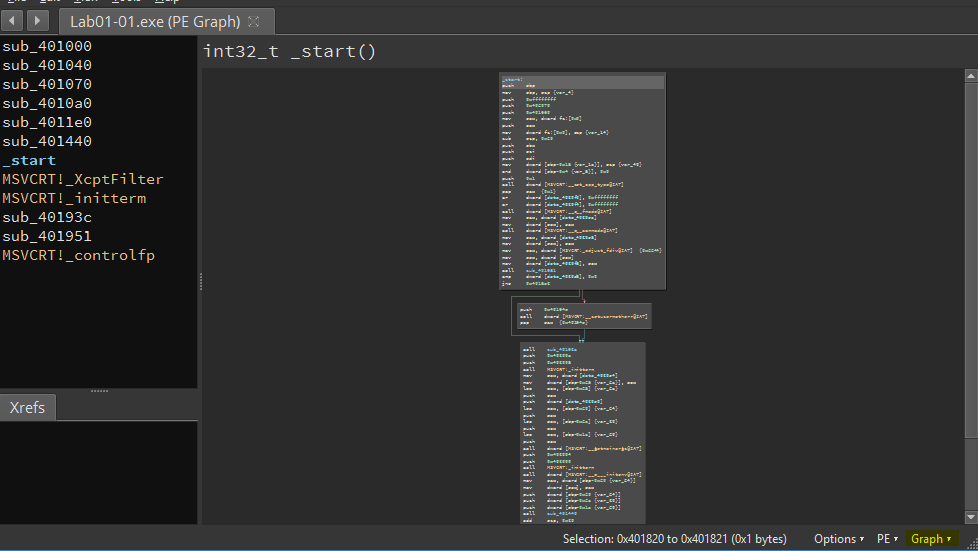
**Figure 2- The default malware analysis screen**

## Disassembly Window Modes

Binary Ninja offers three primary modes to view disassembled malware. The default mode is graph, with the alternate options of linear and hex mode. Switch between the graph and linear modes by pressing **spacebar.** Switch from either of these modes to hex mode by pressing the **h** key. In order to switch from hex back to the graph or linear, the **h** key must be pressed again. The strings view may also be accessed by clicking on the bottom-right drop down menu and selecting Strings. This is the highlighted dropdown in figure 3. **Spacebar** and **h** are the recommended methods to switch modes but the **right-click** context menu or highlighted section in figure 3 may be used as well if these shortcuts are forgotten.

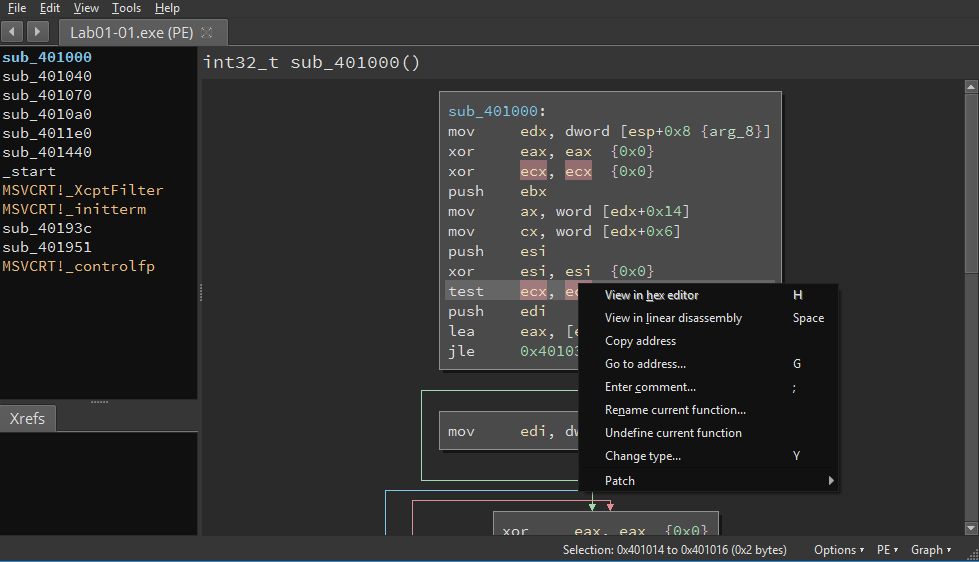
### Graph Mode

Graph mode has a variety of features that are useful for gaining an overview of the specimen that is being analyzed. A recommended first step for graph view is to press the options tab in the bottom-right of the window and enable **Show opcode bytes** and **Show address**. Blocks of assembly have multi-colored lines connecting them that signify jumps in the code. A red-line jump indicates that the jump is taken if the condition is not met. A blue-line jump indicates an unconditional jump. A green-line jump indicates that if the condition for the jump is met the path will be taken.



**Figure 3- Default graph view**

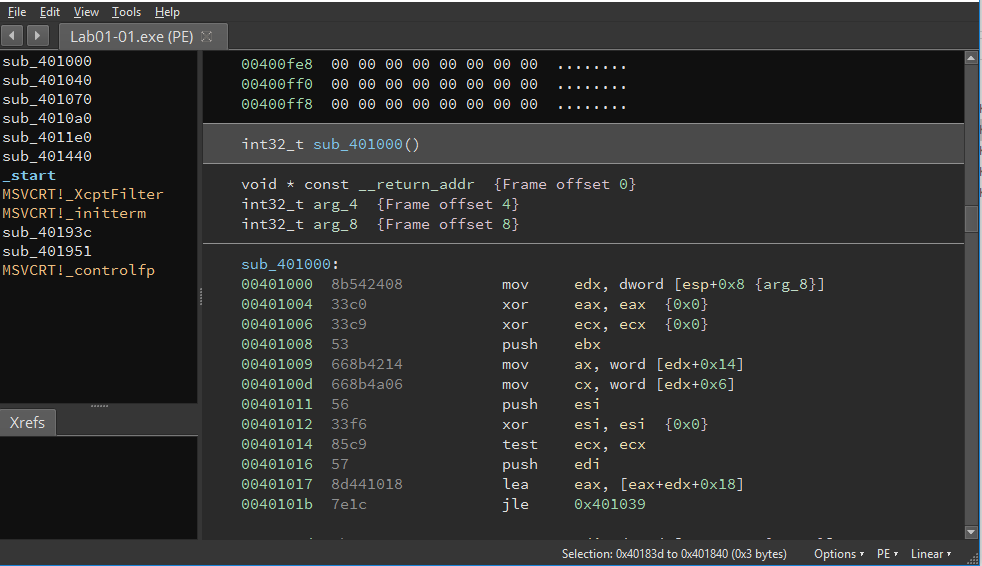
By **double-clicking** the jump-lines, the block that the jump references will be centered and highlighted.The **mouse wheel** is used to zoom in and out. **Right-click** will open a context menu that changes based on the location of the click as shown in figure 4. These features will be discussed further in a later section.



**Figure 4- Right-click context menu**

### Linear Mode

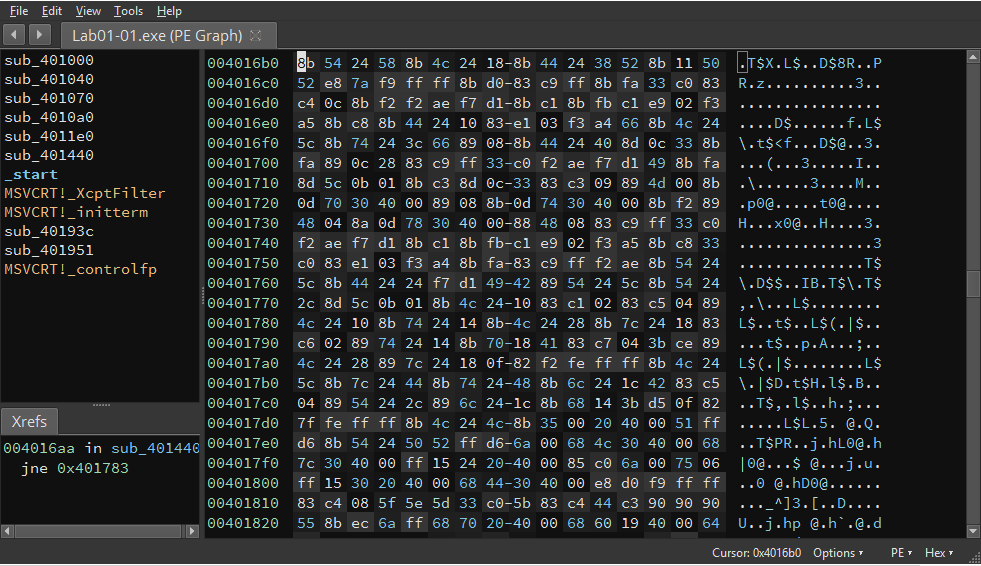
While graph mode provides an excellent overview of the control flow of a program, sometimes a more linear presentation of code is easier to understand. When switching to linear mode, the cursor will stay on the currently highlighted line and center the screen based on the current position in graph mode. Linear view is useful as a “combined” graph and raw-hex view. It can allow one to find sections that were not properly identified as code. This view is also useful for identifying and adding type information for unknown data. Many reverse engineers will switch dynamically between the more “big-picture” graph mode and the sometimes “easy-to-read” linear mode.



**Figure 5- Default linear view**

### Hex Editor Mode

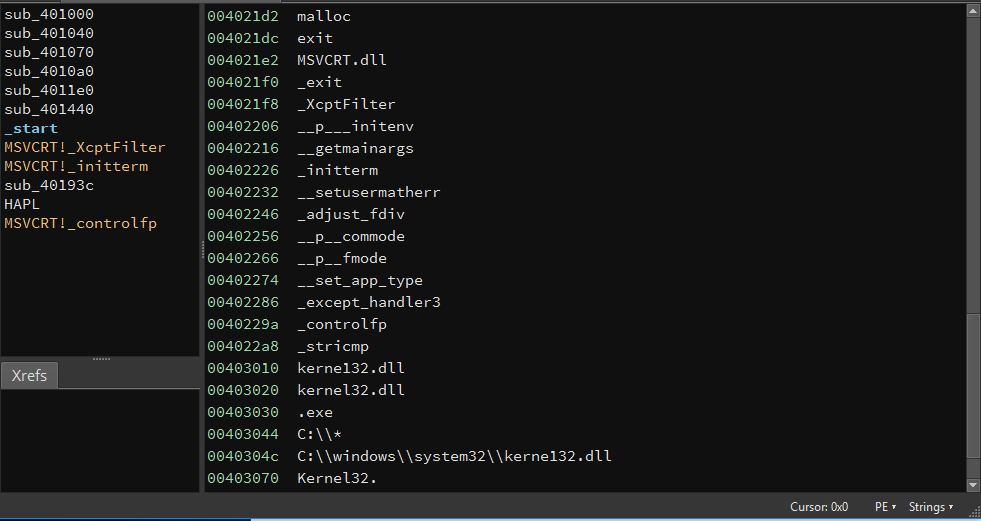
The hex editor mode is used to view the contents of the raw binary file. It can be used to transform data dynamically and see the results of these changes in the disassembler. The uses for these transformations are discussed in a later section.



**Figure 6- Default hex editor view**

### Strings Mode

Use the dropdown menu in the bottom-right to select Strings view. This mode shows all detected strings and their addresses in a sorted list. On can use this view to find keywords or phrases that could be useful indicators of possible program behavior. A handy shortcut for this view is **ctrl+f** which lets you find substrings.



**Figure 7- Strings mode**

## 

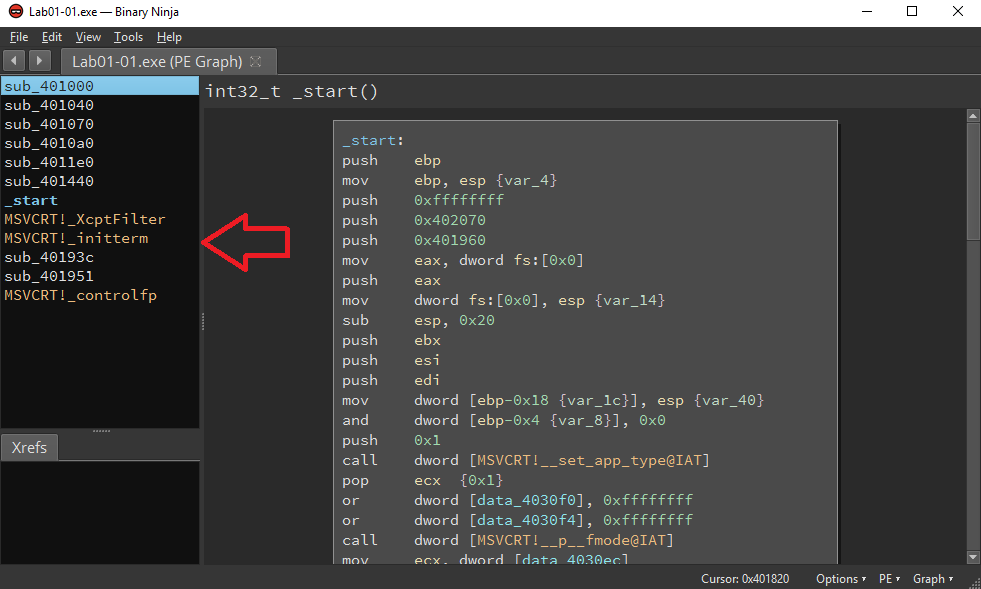
## Persistent Windows

There are also a couple of windows that carry over across different views. The function window and the cross reference window are both visible in all three view modes. These windows provide useful information about the specimen that can be useful during all stages of analysis.

### 

### Functions List

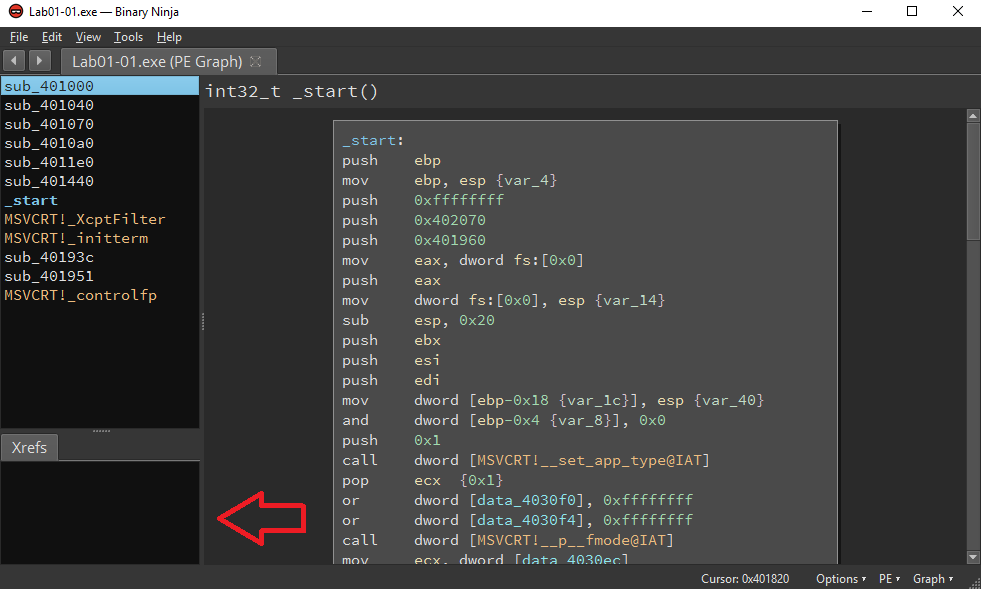
The functions list, as shown in figure 7, shows all functions identified by the disassembler. This list is dynamic and it may grow during analysis. This list can be used as a reference for known functions. When combined with the ability to manually rename functions, the list can be an extremely useful tool for keeping track of subroutines and imports. Note that imports are listed in a highlighted font (orange by default) and certain keywords will be a bold blue font.



**Figure 8- Functions list**

### Cross References (Xrefs) View

Binary Ninja also offers a cross references (xrefs) view at all times. This window, located in the lower-left portion of the screen, offers a glimpse of all cross-references to a given location or reference. The cross-references shown here will change based on the selection, so whatever is highlighted with the cursor will be referred to here. A useful trick with the cross-reference view is to use the hex editor to select a large range of memory addresses to see all the cross-references to addresses within that range.



**Figure 9- Xrefs**

## Context Menu Features

Binary Ninja includes a context menu that can be accessed via a **right-click** of the mouse or pointing device. Figure 4 shows this context menu in the graph view when used on a specific line. This section covers features in the right-click context menu.

* **View in hex editor -** opens the current file in hex editor view
* **View in linear disassembly-** opens the current file in linear disassembly view
* **Copy Address -** copy selected address to the clipboard
* **Go to address -** go to the address entered by the user if it exists
* **Enter Comment** - enter a comment for personal use
* **Rename Symbol -** rename a symbol for later reference
* **Create function at symbol** - another form of renaming, creates a function at current selection
* **Rename current function** - rename the currently selected function for later reference
* **Undefine current function** - removes a function definition, useful for removing erroneous definitions
* **Change type…** - changes the type of selected item
* **Display as… -** change the current displayed value from the default to a wide range including binary, hex, etc....
* **Patch -** perform a variety of operations that affect the code such as NOP.
  + NOTE: Use **ctrl-z** to undo any changes
* Certain items may offer a highlight option to easily track instructions or lines.

## Keyboard Shortcuts

The following shortcuts are useful when performing disassembly. These functions are largely identical to those found in the context menu with added ease-of-access. The following are listed in the Binary Ninja getting started guide:

* **h** Switch to hex view
* **p** Create a function
* **g** Go-to-address dialog
* **n** Name a symbol
* **u** Undefine a symbol
* **e** Edit an instruction (by modifying the original binary -- currently only enabled for x86, and x64)
* **x** Focus the cross-reference pane
* **;** Add a comment
* **i** Switch between disassembly and low-level IL in graph view
* **y** Change type
* **d** Switch between data variables of various widths
* **r** Change the data type to single ASCII character
* **o** Create a pointer data type