VERA Version 0.31
User Manual and Documentation

Danny Quist, Ph.D.
Los Alamos National Laboratory
Advanced Computing Solutions

August 18, 2011

Contents

1 Introduction **1**
1.1 Recommended Hardware 1
1.2 License and Copyright Information 1

2 Installing VERA **2**

3 Ether **2**
3.1 Installation Steps 2
3.2 Setting up a Windows Virtual Machine 3
3.3 Generating Traces 3
3.4 Generating the Traces 3
3.5 Trace File Format 4

4 Intel PIN-Based VERAtrace **5**

5 VERA GUI and Usage **6**
5.1 Loading a Sample Trace File 6
5.2 Interacting with the Graph 7
5.3 Identifiying Program Constructs in VERA 8
5.4 Interacting with IDA Pro 9

1 Introduction

VERA is a visualization tool for analyzing compiled executables. It is built on an OpenGL framework with the wxWidgets package. The current version is only for use with Windows XP and higher operating systems. This manual will detail the steps that are needed to run and analyze a sample of malware.

There are two ways to generate trace data for VERA. The first is with the Ether hypervisor. Ether is a set of patches made to the Xen hypervisor that allows for covert analysis of running processes. It makes an ideal environment to monitor and trace running programs. More information is available from the Ether website.[1] The next option is to use the VERAtrace Intel PIN module. This is a much simpler way of running traces and can be used inside any virtual machine. When available, choose the Ether system for generating traces. Ether is more resilient to detection over the Intel PIN-based VERAtrace.

1.1 Recommended Hardware

VERA is implemented using the OpenGL system with the wxWidgets API. In order to get the best results out of VERA, we strongly recommend you run it on a machine with a hardware graphics accelerator. The code was developed using an Nvidia GTX 285 and subsequently tested on a variety of other cards.

1.2 License and Copyright Information

[1] Ether Website: `http://ether.gtisc.gtech.edu`

INFORMATION, APPARATUS, PRODUCT, OR PROCESS DISCLOSED, OR REPRESENTS THAT ITS USE WOULD NOT INFRINGE PRIVATELY OWNED RIGHTS.

2 Installing VERA

Installing VERA is very straightforward. Simply download the package from Offensive Computing[2] and double-click on the installation file. The files will be installed into the standard Program Files directory unless you specify otherwise.

To execute VERA either find the VERA directory in program files and double-click the wxvera.exe file, or find the shortcut that was placed in the Start menu or on the desktop.

3 Ether

Installing the Ether patches to Xen can be an interesting experience. This section will attempt to aide you in this process. As always please consult with the official Georgia Tech Ether website for the most up-to-date information.

3.1 Installation Steps

There are some general steps to install an Ether system. Most of the problems that many people have are related to trying to compile the source from scratch. To make this process easier we have provided a precompiled Debian package. This alleviates many of the problems that most are having with the installation. Following these steps exactly will get you through much of the difficulty.

1. Download and install the Debian AMD64 net installation ISO.[3] You'll need to get the "Lenny" release of Debian. Also make sure to get the 64-bit version installed.

2. Install the required packages for a working Xen system. A complete list can be found at Offensive Computing.[4]

3. Next install the Ether system. There are two methods for doing this: from source or from a Debian package. We have prepared a Debian package that may alleviate some of the problems of compiling from source.[5]

[2] http://www.offensivecomputing.net/vera/
[3] http://www.debian.org/CD/netinst/
[4] http://www.offensivecomputing.net/ether_install_packages.log
[5] http://www.offensivecomputing.net/?q=node/1575

4. Make sure that the Grub configuration matches your system installation. While the package we have prepared does a decent job of preparing the menu.lst file, there are some problems that may arise. Figure 1 has an example of a working configuration file.

5. Reboot and verify that your new Xen/Ether system is up and running.

```
title Debian GNU/Linux, kernel 2.6.26-2-xen-amd64
root (hd0,0)
kernel /boot/xen-3.1.0.gz dom0_mem=1G
module /boot/vmlinuz-2.6.26-2-xen-amd64 root=/dev/sda1 ro quiet
module /boot/initrd.img-2.6.26-2-xen-amd64
```

Figure 1: An example GRUB menu.lst file for a working Ether installation

3.2 Setting up a Windows Virtual Machine

Creating a virtual machine after Ether is installed can be problematic. There is a bug that will freeze the installation during the install program's execution. To get around this problem, simply boot into a non-Ether patched system when you are first configuring a VM.

Ensure that you have followed the VM installation instructions from Georgia Tech.[6] Failure to create a proper VM will result in crashes and other problems. Once your system is installed, you can begin taking traces for use inside of VERA.

3.3 Generating Traces

The primary unit of data that VERA operates on is a trace file. The format for the traces is the output from the Ether "instrtrace" command. The output generally contains a listing of addresses and the associated instruction that is executed. VERA loads this trace file, processes it, then outputs a graph markup language (GML) formatted file. VERA can display GML files without any additional processing.

3.4 Generating the Traces

This section will overview generating traces inside of Ether. The example that will be used is the notepad.exe file. First, start up a virtual machine inside of the Ether

[6] `http://ether.gtisc.gatech.edu/xen_install_windows.pdf`

system. Once the OS is booted, the virtual machine ID will be needed. To find the ID, simply run the “xm list” command. This will display a listing of running virtual machines. This ID is necessary to run Ether.

Next execute ether using the following command:

```
ether_ctl instrace # notepad.exe > notepad.trace
```

Be sure to substitute the “#” with the VM ID from the “xm list” command. This will generate a text listing of some initial Ether boilerplate, and the instruction traces used to build the later GML file. Transfer the notepad.trace file to the analysis machine where VERA is installed. Section 5 details loading the trace files in the GUI.

3.5 Trace File Format

If you would like to generate your own trace files for use in VERA, they must have a specific format matching the Ether instruction traces. The beginning of the file should match the output from Figure 2.

```
After init:
        shared_page_ptr: 0xffff830000fd9000
        shared_page_mfn: 0xfd9
        domid_source: 0
        event_channel_port: 34
Shared Page va: 0x7fde19b77000
Shared Page test:
        Page-Sharing is A-OK!

Trying to bind to local port...
Success, bound to local port: 35
Trying to get first pending notification...
Taking off suprious pending notification...
Setting filter by name to: notepad.exe
Execution of Target detected:
        Image Base:  0x1000000
        Image Size:  0x14000
        Entry Point: 0x100739d
```

Figure 2: The starting boilerplate for an instruction trace.

Next will be the instruction traces. As shown in Figure 3, the format is a hex virtual address, that should match the contents of the executable, and the ASCII representation of the instruction.

```
100739d: push   0x70
100739d: push   0x70
100739f: push   0x01001898
10073a4: call   0x01007568
1007568: push   0x010075BA
100756d: mov    eax, fs:[0x00000000]
1007573: push   eax
```

Figure 3: The contents of the instruction trace portion.

Finally, the end of the file should contain the string "Handling sigint" twice. This denotes the end of a trace. Figure 4 shows an example of this data. Once this is completed, the files should be parsable by VERA and will generate graphs.

```
1007519: jnz 0x01007522
100751b: push esi
100751c: call [0x1001318]
Handling sigint
Handling sigint
```

Figure 4: The ending structure of the trace file.

A full sample file can be found on the VERA webpage.[7]

4 Intel PIN-Based VERAtrace

VERAtrace is a tracing system based on the Intel PIN-monitoring framework. It can be used to generate similar traces as can be found from Ether, but does not require a full Xen/Ether installation. Traces also include an import resolution system, which can be displayed in the VERA GUI.

To generate traces, you will need the following files from the VERA installation directory:

- pin.exe

[7] http://www.offensivecomputing.net/vera/notepad.trace.gz

- pinvm.dll
- taipin.dll
- veratrace.dll

The above files may be copied to any virtual machine as long as it is running in 32-bit mode. To run a trace, simply execute the command listed in Figure 5

```
pin.exe -t veratrace -- \\windows\\system32\\sol.exe
```

Figure 5: Command line execution of veratrace of the Windows XP Solitaire program

The output will be a trace file with the name of the executable (minus the directory) along with the PID used to run the program. The output of the command in Figure 5 was the file named “sol.exe-2015.trace”. This file can then be copied to the VERA directory and processed normally.

5 VERA GUI and Usage

The VERA GUI is a front-end to allow you to quickly visualize and explore a program’s execution trace. After generating a trace file, VERA can be used to process and generate a graph. This process converts the trace into a GML file that can be loaded and explored in the GUI.

5.1 Loading a Sample Trace File

Included in the VERA executable distribution is a sample trace file that contains a runtime trace of the “notepad.exe” program from a standard Windows XP Service Pack 2 installation. To load the program, simply use the open folder icon in the tool bar or the “File/Open” menus; find the trace file named “notepad.trace”.

A separate window will open, resembling Figure 6. Two pieces of information must be entered in these fields. The first is the original executable for the file. This file is needed to analyze the running sections of the executable to color the graph. The second piece of information is the name to use to store the GML file. The name you enter here will be the source for the output names based on the type of graph you’re looking for. The next options are for processing a graph that has all the addresses rendered as a node in the graph. Rendering all the addresses as vertices is referred to as “All Addresses” mode. To only render the beginnings and ends of

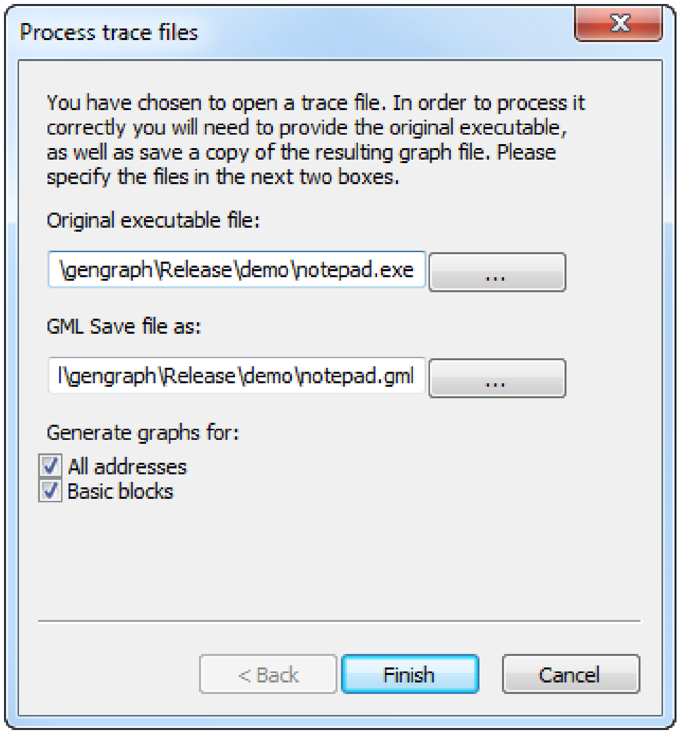

Figure 6: The dialog for processing traces within VERA.

basic blocks, select the "Basic Block" check box. Once you are ready to process the trace file, click the "Finish" button. A new thread will be created to show you the results of the graph. Figure 7 shows the results of parsing the Notepad trace.

Depending on the options, two different graphs can be created. The first will be the all vertices graph. This will have the name of "all-yourgraphname.gml". The second is the basic blocks graph. This will have the name of "bbl-yourgraphname.gml". The default behavior is for VERA to load the "all-" file. To load the basic blocks graph, load the "bbl-" file using the "File/Open" menus.

5.2 Interacting with the Graph

Interaction with the generated graphs was designed to emulate many common 2D navigation systems. The interface is similar to that of Google Maps. There are two methods for moving a graph: First by panning left, right, up, and down. The second is to zoom in and out. All are accomplished with the mouse. Table 1 lists each of the controls for interacting with the graph.

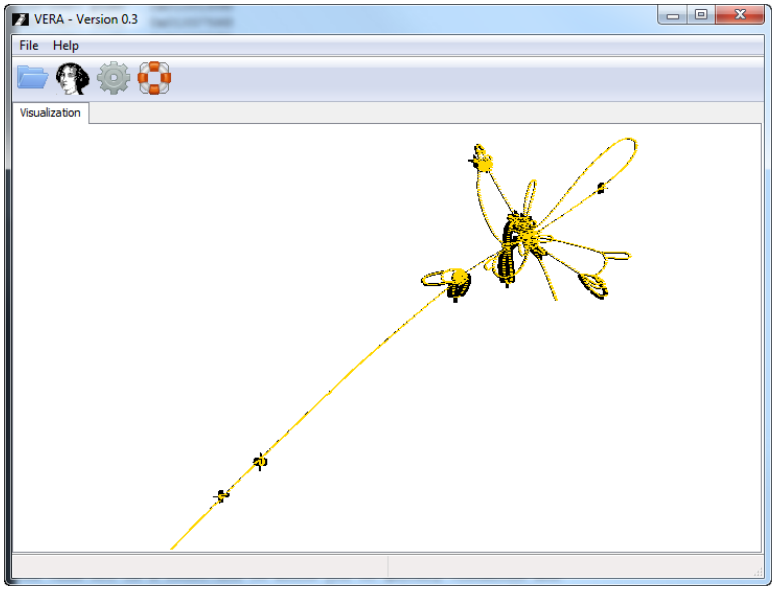

Figure 7: The resulting graph from processing the Notepad trace.

Action	Mouse Control
Pan Left	Left-click, drag left
Pan Right	Left-click, drag right
Zoom In	Mouse wheel up
Zoom Out	Mouse wheel down
Navigate in IDA	Right-click

Table 1: Mouse controls for navigation.

5.3 Identifiying Program Constructs in VERA

Various features in the graph correlate to features inside of the code. First, any series of addresses that have exactly one entrance and one exit, and that are not part of a loop, can be safely considered initialization code. Many of the constructs also show up in multiple areas of the executable. The initialization portion of the Notepad graph in Figure 7 shows an initial series of instructions, followed by a loop prior to entering the main portion of execution. This is a very common shape created by Microsoft compilers.

A section of code that shows a branching operation, such as illustrated in Figure 8, correlate to a decision point in the program. Please note that if the branch always takes a single path, the structure will not be shown in the graph. This is because VERA relies upon traces of execution from the code, and does not note the possible paths of execution available.

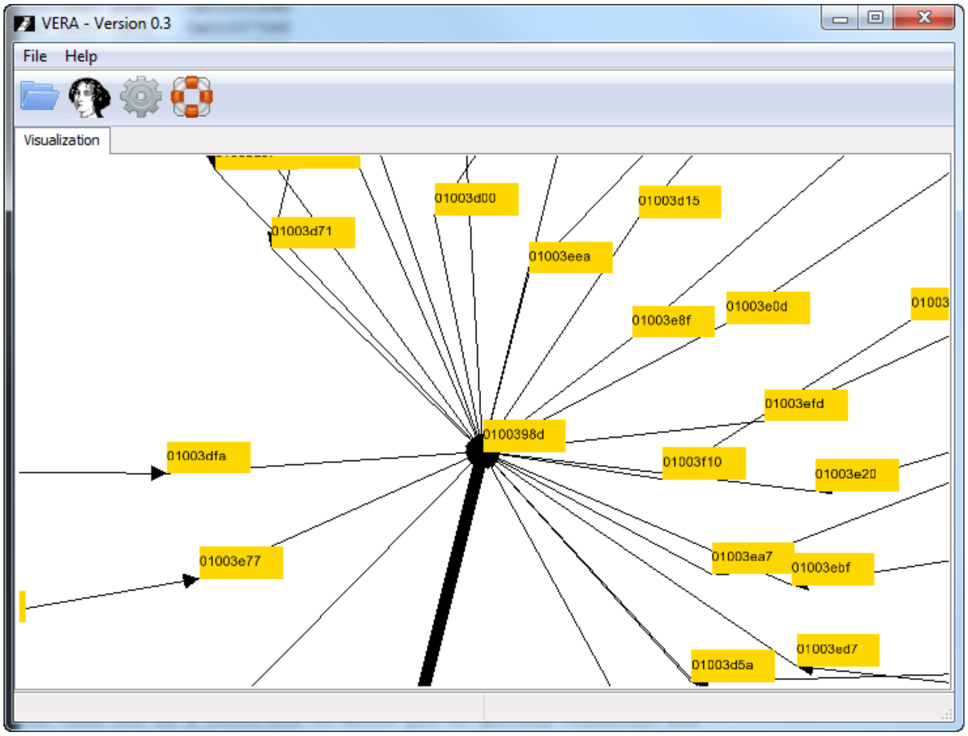

Figure 8: A series of branching operations from the Notepad.exe program.

Whenever a portion of code is executed multiple times, the edges between the nodes are drawn with a thicker width. This allows for quicker identification of the most commonly used areas of code. For instance, if a messaging processing loop is present in a program, that loop will be highlighted. Figure 9 shows a loop in the Notepad program.

5.4 Interacting with IDA Pro

A new feature in VERA 0.3 is the interaction capability with a program loaded in IDA Pro. This alleviates the need to separately interact with IDA and manually enter the addresses. To initiate a connection to an IDA system, you will need a couple of things: First, of course, is a copy of IDA Pro. The second is the VERA–IDA plugin for successful interaction with IDA. You can download a copy from

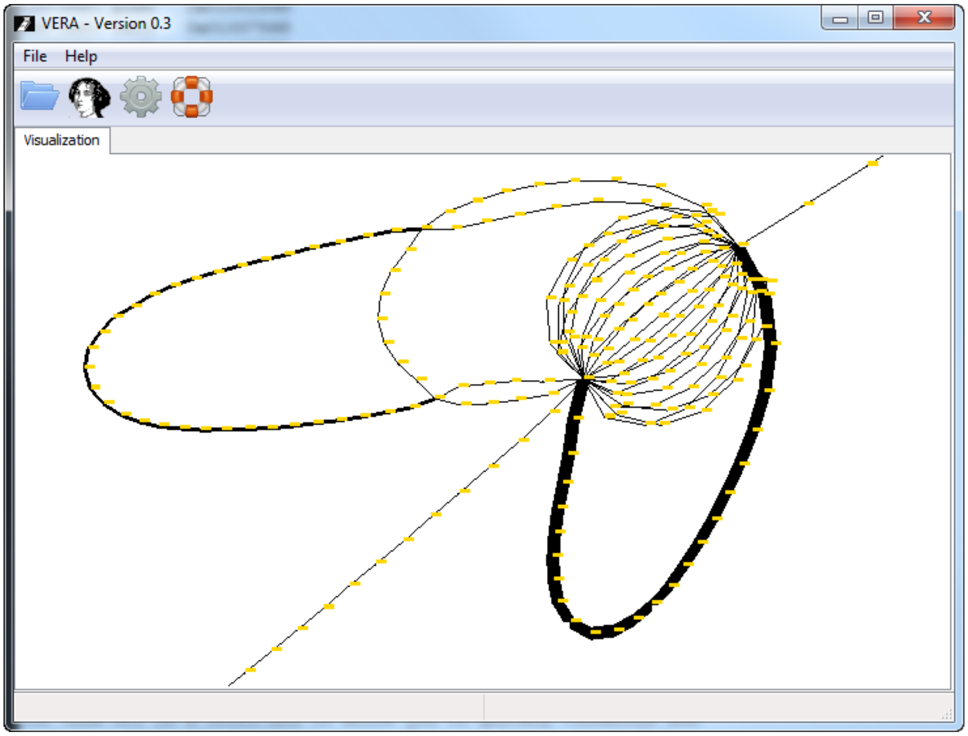

Figure 9: A loop with multiple branches inside the Notepad.exe program.

Offensive Computing.[8]

Precompiled versions of the IDA module are available for IDA Pro versions 5.6 and later. The source code is available for earlier versions of IDA, and you're welcome to try to compile it. They are not, however, officially supported.

The plugin should be copied into the "plugins" directory of the IDA Pro installation directory. Once this has been done, you will need to restart IDA.

Once the module has been installed and IDA has been restarted, you should see a message in your IDA console that looks very similar to Figure 11. If you do not see the message, then you will need to verify that the IDA plugin was installed correctly.

To start the IDA Pro server, click on the IDA Pro icon in the VERA toolbar. From here the server should be started. Next, you will need to figure out the computer's IP address. A TCP/IP connection is made from the IDA Pro system to the VERA system. This allows you to take advantage of virtual machines when analyzing malware. This architecture was chosen to allow for an IDA Pro instance on a

[8] `http://www.offensivecomputing.net/vera`

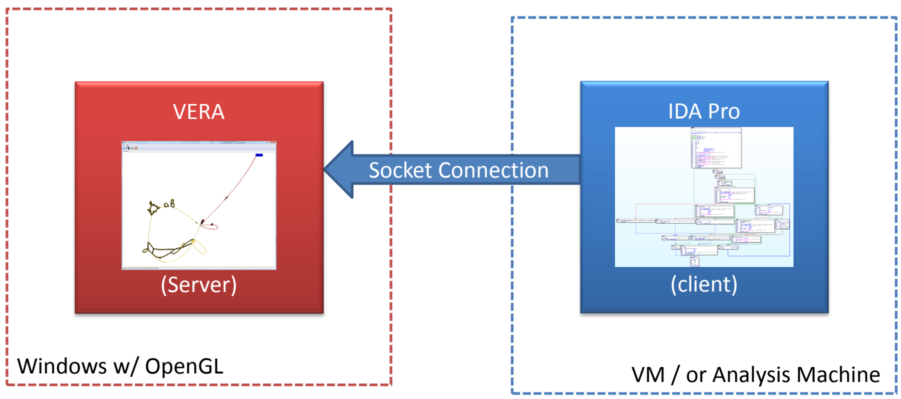

Figure 10: Architecture of the VERA IDA Plugin

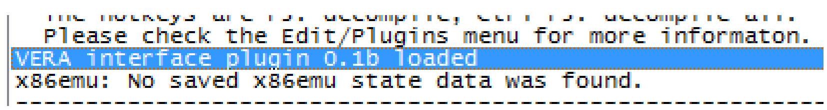

Figure 11: IDA console text showing the IDA Pro module loaded correctly.

virtual machine talk to VERA running on a machine with a real hardware graphics accelerator.