Visual Toolkit for Network Security Experiment Specification and Data Analysis*

L. Li
College of IST
Penn State University
lunquan@psu.edu

P. Liu
College of IST
Penn State University
pliu@ist.psu.edu

G. Kesidis
CSE Department
Penn State University
kesidis@engr.psu.edu

ABSTRACT

The increasing availability of network testbeds and the benefits of visualization-based security study call for the emergence of supporting tools for network security research. In this article we present ESVT, an integrated experiment specification and visualization toolkit that supports network experimenters to conduct interactive experiments on network testbeds such as DETER and Emulab. The ESVT package includes a topology builder including experiment specification, a TCL script generator, and various visualization tools. The unique feature of ESVT visualization is the combination of topology-based network animation for global awareness and detailed data analysis support through a complete set of data conversion, data selection, and graphical analytical tools.

Categories and Subject Descriptors

C.2.0 [**Computer and Communication Networks**]: General: security and protection; I.3.8 [**Computer Graphics**]: Applications; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces

General Terms

Experimentation, Security

Keywords

Animated topology, testbed, experiment specification, visual toolkit.

1. INTRODUCTION

Network testbeds provide a flexible environment with which researchers can configure and construct arbitrary network topologies and test new protocols or security defense technologies on them using simulation, emulation, and hybrid experimental approaches. Among various network testbeds, the clustered, time-shared, close-controlled, and fully segregated (among experiments) testbeds such as Emulab [3] and DETER [2] have gained momentum and their scales are increasing as well. The currently deployed DETER testbed has close to 300 experimental nodes and Emulab has more than 300 nodes. As stated in their design requirement document, the Emulab and DETER testbeds have the following features that will benefit network and security research: realistic network topology; non-production network; support for repeatable experiments; rapid reconfiguration; realistic network traffic; data archiving; network management, monitoring, and analysis; physical and network access control; fidelity; programmability. A number of security research experiments including DDoS attack and defense, worm propagation and defense, and defense of enterprise networks are being conducted on those testbeds.

To conduct network experiments, especially a network experiment on a testbed, the following experimental components are normally required.

⊙ Network physical topology - Network packets flowing on a network are routed and constrained by the underlying network topology, so the first step of a network experiment is to define or design a topology. Experimenters need to decide the number of end-host nodes, server nodes, router or switch nodes, sub-network or Internet virtual nodes, the layout of those nodes and connecting links between those nodes, and network properties such as the bandwidth and latency on all those elements.

⊙ Traffic generator - Most network experimenters are interested in the traffic flowing on the network and their patterns. An realistic network experiment must have adequate background traffic generated from experiment nodes. The traffic can be generated based on mathematical models, or by replaying collected traffic logs from real world networks.

⊙ Traffic logger - Network traffic has to be collected for later analysis. Various traffic capturing tools are available for this purpose, among which TCPDUMP is the most popular one. Most simulation applications also have such functionalities built-in in the package.

⊙ Traffic analysis or visualization - Analytical tools can support such operations as format conversion, filtering, reordering, digestion (clustering and categorizing), and data mining. Recently, visualization has been found to be highly effective in inspecting and analyzing network traffic and many visualization tools are developed.

*This research was funded by the NSF and the DHS under NSF grant ANI-0335241, see http://emist.isi.psu.edu The corresponding author for this paper is L. Li, lunquan@psu.edu

VizSEC'06, November 3, 2006, Alexandria, Virginia, USA.

Though there are tools available for each of those experimental components or operations, they are either disjointed or not compatible with each other, or not user-friendly enough for an interactive experimental environment. Additionally for a testbed experiment, a TCL script for the experiment specification is required, which is not easy to write and maintain if the scale of topology becomes large. For the benefit of network experimenters, an integrated experiment tool suite is desired, which can support topology configuration, experiment specification, TCL script generation, and traffic visualization.

For such purpose, the EMIST [4] team has developed the ESVT graphical user interface, or Experiment Specification and Visualization Tool, which provides an integrated environment to interact with DETER or Emulab test-beds and to conduct network security emulation/simulation experiments. More importantly, the ESVT GUI uses animated topology and multiple visualization means to replay experimental results and help experimenters run an interactive and visual data analysis, greatly improving the efficiency of data sense-making and problem diagnosis. Finally, the ESVT can manage experimental data captured in TCPDUMP or flow format using relational MySQL database in a unified way, enabling easy data integration and more advanced data analysis.

2. RELATED TOOLS AND CHALLENGES WITH TESTBED OPERATION

Researchers have developed various tools to help designing network experiments, including topology generators, traffic collectors, traffic analyzers and visualization tools.

2.1 Topology Tools

The choice of topology tools is dependent on the scale of the target experiment network. If the network is small, experimenters can choose to write a script to build and configure the network manually. If the target network is large or the whole Internet, topology generation tools can be helpful or necessary. Those tools include the Inet topology generator from University of Michigan which generates global-scale network topology, the GT-ITM topology generator from Georgia Tech which creates flat random graphs or N-level hierarchical graphs, Brite topology generator from Boston University, and tiers topology generator. [1]

The Emulab and DETER testbeds also supply a GUI topology tool for testbed users to design a small or medium scale network. The advantage of this tool over other topology tools is that experimenters can both build the topology and interact with testbed nodes using the same interface.

However, those topology tools either have no graphical user interface for user-friendly interaction, or are not suitable for designing large-scale topologies such as enterprise networks which may include hundreds or thousands of nodes. Additionally, neither of these tools can support script generation with topology scaling-down by virtualization, which is necessary to run an enterprise network emulation or simulation experiment on a testbed. The script generator in the ESVT supports LAN virtualization using either Emulab VM or the EMIST virtual node approach.

[1]The links for these tools can be found on the NS web site.

2.2 Traffic Collector, Analyzer and Visualization Tools

These tools include traffic capture programs, trace format translators, trace filter programs, digestion tools, data mining programs, traffic visualization packages, etc. A number of tools can be found from the TCPDUMP [6] web site, such as Ethereal, CoralReef, Tcptrace, and Sniff. Generic mathematical and plotting packages such as Matlab, Gnuplot, and RRDTool can be used as well in analyzing and visualizing network experimental results.

The tools just mentioned are mainly for the network traffic on one specific link or network interface. To get a sense of overall network dynamics, a different kind of visualization tool is needed, which can present the snapshot of the whole network in one screen, and change the display as the time elapses. Such global animation tools can increase the situational awareness of the network dynamics and events to the experimenters. NVisionIP [1] from NCSA is one example of such tools. However, the visualization of NvisionIP tool is indexed by IP address, not by the network topology.

The NS/2 simulation package also has a visualization tool called NAM [5] that uses packet-level animation to replay network simulation details. The packet symbols are flowing on the underlying topology so the user can directly sense the network dynamics and find the possible bottlenecks or problems. The time scale suitable for NAM visualization is normally very short, at millisecond and lower levels.

Animation and interactive view generation are widely used in real-time network traffic and traffic log data visualization [9, 13]. The ESVT visualization module combines the advantages of these visualization tools. It is comprised of a global topology-based animation, detailed link traffic charts, and user-defined flow visualization views.

Visualization tools are not just for network traffic data. A number of visualization tools and supporting scripts have been developed in visualizing BGP data [11], machine status data [12], and DDoS behaviors [8].

The rest of paper is organized by the following order. In the next section we introduce the topology function of the ESVT. Following that is the section which discusses of using the ESVT to generate TCL scripts with various of layout or scale-down options and start-up program setting for experiment nodes. In section 4 and section 5 we present the visualization part of tools and the details of user-defined traffic analysis utilities.

3. AN INTERACTIVE AND VISUAL EXPERIMENT BUILDER

The ESVT is a modular, component-based topology editor, a TCL script generator, a worm experiment designer, and a visualization tool for experimental results. First, the tools offer a topology editor toolbar for users to draw network topologies including computer/end-host nodes, switch nodes, router nodes, sub-network/Internet interfaces and links. Each of these network components has configurable properties such as bandwidth and link latency, which can be stipulated and modified conveniently by the graphical interface. Alternately, users can import a scale-free topology generated by the GT-ITM tool into the ESVT editor. The ESVT can then generate a TCL script from the completed network topology in several formats for NS simulation or DETER/Emulab emulation.

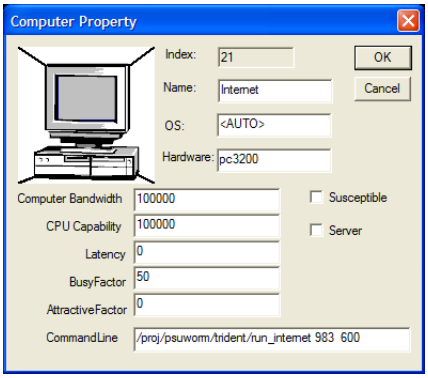

Figure 1: Change component properties.

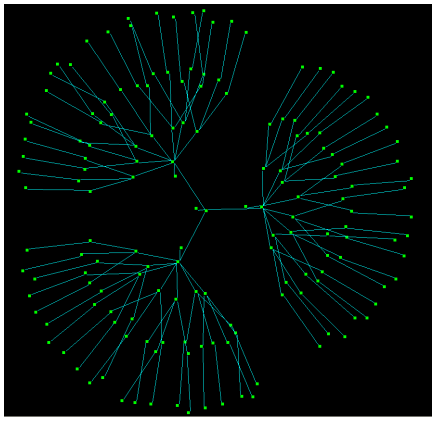

Figure 2: Import topologies from NS2/TCL script files.

3.1 Basic Topology Editing

The ESVT GUI provides an integrated environment to plan and specify interactive network experiments. At the first step, it can be used to draw the network topology. The toolbox of programs includes network components such as computer/host node, switch, router, network/Internet interface, and link. Each component has a number of properties that can be modified in-place. Selecting the component and right-clicking on it will open the property dialog window, as seen in Figure 1. Users can also change the properties of a group of components or all components by invoking the global component and script property configuration window. The globally changeable properties include host susceptibility ratio, link bandwidth, and experiment duration time.

The topology builder is designed to be scalable and can support large topologies with thousands of components. There is no hard limit on the size of the virtual canvas and the display can be smoothly zoomed in and zoomed out to commodate editing operation on a large topology.

A special feature of ESVT topology builder is its support for experiment virtualization configuration. To emulate network dynamics of a large topology on the resource-limited testbed, experimenters need to design or adopt some kind of scale-down or virtualization approach to translate the original topology into a scaled-down topology that can be run on the testbed. The method of virtualization support in the ESVT topology builder is through sub-LAN or switch marking. To specify one sub-LAN to be emulated or simulated using a virtualization approach, experimenters can change the property of the stub switch node in that network segment. The switch node has one special property called *Simulated LAN* and modifying the value of this property will change the way the ESVT generates the DETER/Emulab TCL script[2]. A virtualized switch is distinguished from a real or non-virtualized switch whose LAN segment will not be scaled-down by its color and symbol.

There are some other useful features in the GUI that can help design a large network topology, including topology zoom, component index display option, and component/node finder function.

3.2 Support for Other Topology Formats

The ESVT topology tool also supports topologies from other topology generators or in other formats. In the example of importing a scale-free topology generated by the GT-ITM topology tool, ESVT reads a GT-ITM topology file and replaces hub nodes with switch nodes and leaf nodes with hosts nodes. The layout of networks is done by a similar algorithm used in the network animation tool NAM. The converted topology may need further editing after the import is done.

Internet worm simulation experimenters [14] have used another homogeneous network topology format in their Internet scale-down worm experiments. The Internet is modeled as a group of enterprise networks with various numbers of vulnerable servers connected by a super switch. ESVT reads the file of this format and generates a two-level hub-spoke topology. For the simulation results based on the scaled-down model, ESVT can also visualize the worm propagation effects in animation and in line chart.

For an existing NS2/TCL script edited manually or by another tool, the ESVT can scan the script and import the embedded topology from it. Then users can choose to use the GRAPHWIZ/NEATO tool to do the layout if the tool is installed on the machine, or choose to let the ESVT to do the layout using the ESVT's internal layout module which employs a similar but slightly different layout algorithm from the NEATO program. Figure 2 shows one topology imported from an existing TCL script file.

3.3 Experiment Related Configurations

Besides utilizing the virtualization feature for LAN segments, experimenters can specify other component properties that may be useful and necessary in conducting network security experiments.

- Internet interface. Experimenters can run a specially designed program on this node to simulate the traffic to and from the Internet. The simplest implementation of such program will be a traffic sink that only receives traffic from other nodes.

- Normal and vulnerable nodes. For worm and other security experiments, host nodes can be specified to be vulnerable or not vulnerable to the specific attacks. Background traffic generators may be run on the normal or non-susceptible nodes and vulnerable nodes can

[2]In our worm emulation experiment, we used one testbed node to simulate a switched LAN marked for virtualization.

run specially designed programs that simulate vulnerable services.

- Bandwidth, latency, addresses, etc. Users can also specify their desired hardware type, operating systems using the ESVT property dialogs.

3.4 Topology Output and Script Generation

To request network resources from an testbed and apply a network topology on it, an NS style TCL script file needs to be submitted to the testbed control plane. For people unfamiliar with the TCL language or specific testbed requirements, writing such a script may not be easy, especially for a large topology or with the needs to repeat the experiment with frequent and minor parameter changes. The ESVT has such features that can output a network topology and related experimental configurations into an NS or DETER TCL script. The basic procedure of topology conversion is that every node (except a virtualized LAN segment) in a GUI topology is mapped to one node in the testbed. Links in the topology will still be links in the resulted TCL file, or will be replaced with TCL LAN-making clauses (*make-lan*) according to different experimental needs.

The ESVT supports experiment script generation in four different output formats: NS, one-to-one testbed format, testbed format with EMIST virtual node scaling-down, and testbed format with VM node scaling-down [10]. The differences among these options are at the handling of LAN/node virtualization. For example, the script generated with the virtual node testbed script option supports topology scaling-down by the EMIST virtual node virtualization. A switch node marked with a "*Simulated LAN*" label and all host nodes connecting with it will become just one virtual node in the resulted TCP script file. Those omitted nodes and their properties will be recorded in a separate file named map.001, which can be loaded during the experiment running time.

The internal index numbers in an ESVT topology start from 0 for each class of components, while in the script file a component is given a universal number starting from 0.[3] The ESVT will automatically translate and do the mapping between these two sets of numbers.

The basic mapping between GUI node names/numbers and test-bed IP addresses can be realized through the testbed /etc/hosts file.[4] In [10], we reported an enterprise emulation experiment that adopted our virtual node scale-down approach. For virtualized nodes in virtual node scale-down experiments, experimenters themselves need to assign IP addresses to the virtual nodes (virtual IP) or threads in the virtual node program and name their log files according to information from the map.001 file.

The script generator also inserts additional experiment configuration scripts, which are translated from user specifications. Those script lines include testbed node OS choices, testbed node start-up programs, etc. Normally, experimenters run programs on testbed nodes, so the script generator writes one line of start-up command for each node, e.g., traffic collecting command for router nodes, and background traffic generator command for normal and non-susceptible host nodes.

[3]The ESVT update will support node name by user-defined strings.

[4]So it is required to keep a copy of /etc/hosts file with other experimental logs for the later analysis and visualization.

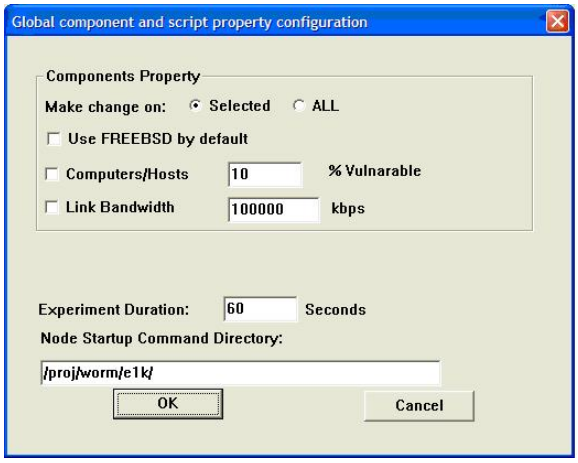

Figure 3: Change the experiment configuration.

Figure 3 shows the *Global component and script property configuration* dialog. In the dialog, there are some parameters that affect the TCL script generation. Clearing the *Node Startup Command Directory* field will disable the start-up command script generation. The "Experiment Duration" value will be one command-line parameter for the start-up commands or scripts, which can be utilized by user applications to control the duration of the experiment. By default, users do not need to specify OS for testbed nodes because the testbed will assign the node operation systems automatically. If *Use FREEBSD by default* checkbox is checked, the script will add a additional line for every node *tb-set-node-os n(##) FBSD-STD*, except for VM nodes which use special *pvcm* OS. A sample of such generated script can be found in appendix A.

4. VISUALIZE EXPERIMENTAL RESULTS USING THE ESVT GUI

Visualization shows clear advantages in helping network and security researchers understand and make sense of the network dynamics from their experimental results and the ESVT tool provides a number of visualization supports for such purposes. There are mainly two categories of experimental data in a testbed experiment: node status log and link traffic dump. The amount of traffic dump data and node status logs (We will use worm infection log as example for node state logs in the following presentation.) after each experimental run can be huge: more than 1,000 files sized at 600-2000 MB for a 60-second experiment of our Slammer worm emulation. The first difficulty is to aggregate these "local" TCPDUMP files and worm infection log files to restore all the events and the "global" causal and temporal relationships among the events. Converting and presenting these data into human readable figures or animations will also be a challenge.

4.1 Animated Topology as Visualization Means

Visualization is fundamentally about events and the partial orders among events. The event could be any anything from packet transfer to computer buffer saturation or over-

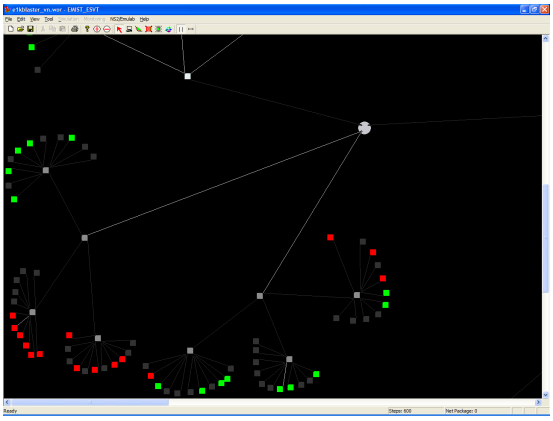

Figure 4: Worm animation: each frame shows current node infection status and link traffic volume

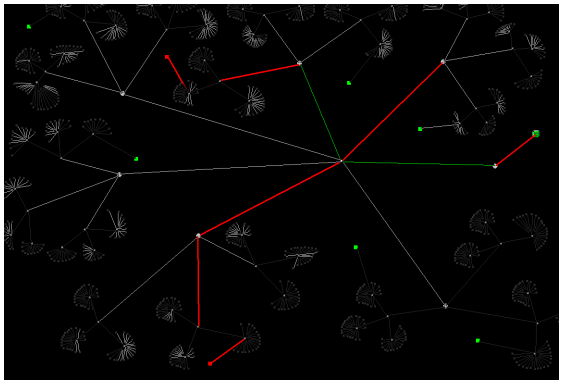

Figure 5: Animated topology for situational awareness

flow. Generally, animation is used to replay network events by time. Some visualization tools like NAM visualize at a very detailed packet level and replay every network event. However such a packet level event-to-event visualization is too complicated for the kind of experiments that are normally conducted on the testbed. Also, packet level visualization may be susceptible to synchronization mistakes. Another way of visualization by showing accumulated effects of worm propagation over a long period of time may miss some important characteristics of the worm. For these reasons, we made a tradeoff and use time-series animation with adjustable step times as the main visualization means. We limit the event types to include only node infection status change, link traffic volume rate, and worm traffic ratio.

The GUI reads network traffic flow and worm infection logs from the experiment log files and uses different animations or histogram charts to replay the worm propagation process and traffic dynamics during the process. The step time of animation can be adjusted from 1 millisecond to 1 minute by the user before the visualization. The program scans all TCPDUMP files and finds the earliest packet time stamp. It then uses this time minus two steps' time as the starting time for the followed calculation and statistics. The infection time of worm victims is calculated by reading the worm infection log file for each infected host. The animation refreshes every one second and the virtual experiment "replay" goes by one step time. Figure 4 is one snapshot of such an animation. Each snapshot is a view that shows the current node infection status and summarizes the average traffic rate during the past time interval.

At every time step, each host node will change its display color to update its current worm infection status. A gray node means the node is not susceptible, a green node means the node is susceptible but not yet infected, and a red one means it is infected. Through this color scheme, the effect of worm propagation can be seen clearly as the number of infected red nodes increases. Traffic dynamics in the network are presented simultaneously by the link color changes. The color of links changes ranging from a gray color that means trivial traffic or below one percent of the link bandwidth during the past time interval, to a red color representing heavy traffic or above thirty percent link bandwidth. If users are interested in the detailed traffic change on one particular link, a bar chart view can be chosen to show the histogram of traffic flow on that link. Animations and bar charts can be paused and fast forwarded or rewound to a desired time to be played or replayed.

4.2 Global Situational Awareness

Through this digestive and integrative arrangement of experimental data and the method of animated topology as the visualization means, the ESVT facilitates users' situational awareness with a holistic presentation of experimental characteristics and many hidden or hard-to-see behaviors of network security problem under study may emerge and be revealed.

Our experience shows that visualization empowers security experimenters with a special and powerful perspective that other methods lack. In the case of SQL Slammer worm emulation we run on the DETER testbed, the salient features of Slammer worm such as bandwidth-saturation scanning and random global scan can be visually seen directly using the ESVT global animation visualization. First, when a susceptible host became infected after receiving scanning traffic from the Internet interface node, the color of the node became red from green. Then instantly after the infection, a red path emerged on the animated topology from the backdrop of indiscriminating traffic, which originated from the infected node to the Internet interface node (The special node which simulates the networks outside the enterprise network under emulation.) through a number of intermediary switch and router nodes. The bandwidth-saturating scanning illustrated by the red path was sustaining and later joined by more red scanning paths, as shown in Figure 5. The characteristics of random and global scans were manifested in that though the scan activities were fast and link saturating, the majority of scanning traffic was directed to the outside of the enterprise, and the local traffic other than around those impacted links experienced little variation. From experimenter's view, such visual characteristics also validated our virtualization emulation framework and the correctness of Slammer emulation program.

The multiple bandwidth-saturating scan flows towards the Internet raised the question of what the combined traffic would be on the egress link. A quick visual check on the enterprise network access link revealed the answer: the aggregate egress traffic actually slowed down as the number of infected hosts increased, which may be counter-intuitive and contrary to many assumed beliefs hold by worm mod-

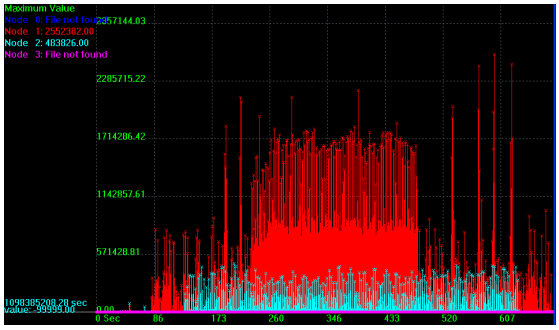

Figure 6: Machine status chart

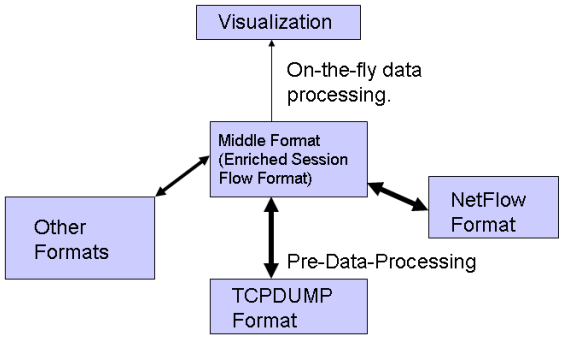

Figure 7: Architecture for User Defined Visualization

Flow_ID	Src IP	Src Port	Dest IP	Dest Port	Start Time	End Time	Protocol

Figure 8: Fields in Traffic Flow Table

elers in the past, but reasonable and making sense when we considered the heavy load on the gateway router and traffic congestion and competition because of multiple worm scans.

4.3 Problem Diagnosis by Visual Inspection

Running emulation and simulation experiments on network testbeds may encounter various problems. Some of problems can be attributed to faulty simulation programs, while others may be because of unfamiliar testbed features and unexpected hardware and network breakdowns. Though the reasons behind those deviant behaviors that raise the doubt of experimenters vary, the process of pinpointing whether the situation is a genuine problem and then possible reason for the problem may be tricky. In this regard, the ESVT can support problem locating and diagnosis through visual inspection of traffic dynamics and testbed machine status.

Here is one example of problem diagnosis of overloaded router node in an emulation experiment conducted on the DETER testbed. On the DETER and Emulab testbed, experimenters have to use a pc node to emulate and take the function of a router that forwards traffic among up to four network interfaces. When multiple hosts were sending heavy and consistent traffic simultaneously through the pc router to a receiving host and the TCPDUMP data collected on the receiving host showed the traffic was erratic and relatively lower than expected, how do we explain this phenomenon and what could be the possible reason for this result? We can suspect the malfunctioning of TCPDUMP program, the overloaded receiving node, or that the pc router is incapable of forwarding multiple high-volume traffic. Through the ESVT's integrated machine status visualization (The machine status data can be captured by the tools developed by Purdue researchers, see [12].), we can select the involved nodes and visually inspect the CPU load and memory utilization of those nodes, as shown in Figure 6. Indeed, the cpu load on the router node is much higher than that on the receiving node and we can further investigate the problem by using a faster computer to replace the current pc router.

Choosing the right information to display on the top of topology can also be useful and important in diagnosing experimental problems. Host infection status and link traffic volume or packet rate are fixed information on the animated topology, users of ESVT can also select additional data to display to inspect and diagnose their interested problems such as IP addresses, node indexes, and detailed traffic readings. In the case of visualizing the worm's sequential scanning behavior after our Blaster worm emulation experiment, we wanted to check whether our emulation program actually scanned locally and sequentially. The visual inspection of the temporal order of color changes on susceptible and infected nodes and the "clustering" effect of worm propagation validated partially that the worm was propagating sequentially. Choosing to display the IP addresses of nodes further verified that the scanning was scanning sequentially, and incrementally in terms of victim IP address sequence.

5. IN-DEPTH DATA ANALYSIS WITH USER-DEFINED VISUALIZATION

Beyond the global topology-based network animation, experimenters may want to go further and analyze the traffic data in a more refined manner. For example, experimenters may want to see the number of session flows in each time interval instead of raw packet numbers or raw throughput; or they want to see only UDP packets during the specific time period. For such detailed data analysis, a user-defined visualization function is provided in the ESVT for link-specific traffic visualization. Users can select a link, define the traffic filter rules such as protocol and time period, check the desired views including histogram, cumulative view, entropy, etc., and the ESVT will run the data analysis and render multiple data views based on the data.

5.1 Unified Data Management

ESVT endeavors to manage experimental trace files in various formats in a unified way. The general framework for the link-specific visualization can be seen in Figure 7. From Figure 7 we can see that the ESVT will automatically convert a TCPDUMP traffic log into a flow-based format similar with the popular NETFLOW, but with detailed packet information for each flow. (The format for the flow table can be seen in Figure 8.) Invisible from the figure 7 is another important feature: the ESVT can store the generated flow data into MYSQL database in two tables: a flow table and a packet table. The use of database to store traffic data has three benefits: a standard way of archiving network experimental results, saving conversion time for the future data analysis, and the opportunity and convenience for network experimenters with database experience to use database analytical tools to study network traffic.

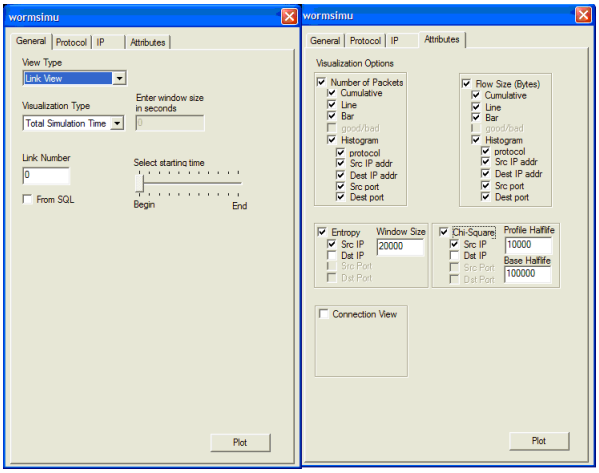

Figure 9: Dialog for link traffic visualization

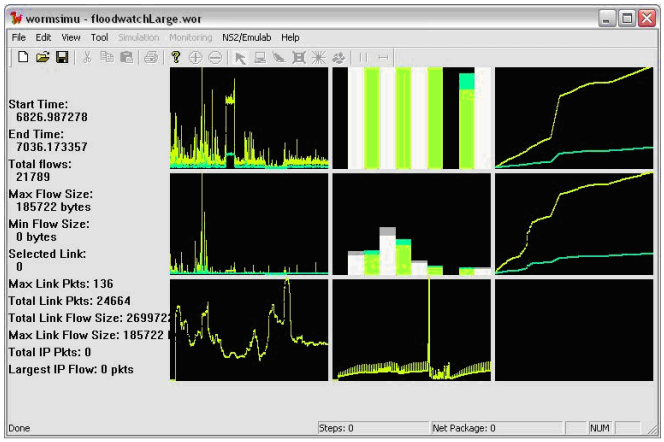

Figure 10: User Defined Visualization: Link traffic

When a user chooses one link to visualize, the ESVT will do the data conversion from the TCPDUMP format to the flow/packet format and store the two tables into the MYSQL database. Then you can either use the resulting raw data source and ESVT data filters to render customized visual presentations, or write advanced SQL statements to combine, select, and manipulate multiple data sources into a new data stream upon which visual rendering can be built. For example to select only those flows whose source IPs appear in link one but not in link two, you can write a SQL statement as follows.

```
Select Linkone.*
    From wormexp_flow_link_one as Linkone
    Where Linkone.src_ip not in
  (select distinct src_ip
   from wormexp_flow_link_two)
   Order by Linkone.Start_time
```

5.2 Interactive View Generation

In link traffic visualization, the focus of interface design is to support data selection and view choices interactively. When the user selects one link for detailed visualization, a dialog box will pop up to let the user specify his/her data selection criteria and the kind of charts he/she wants to use for the visualization. Figure 9 shows what the dialog box looks like. The tabs across the top of the window allow the user to select different options, attributes, and view types. The visualization results of multiple plots are displayed in the main window. Clicking on a single plot will zoom-in and enlarge the plot window and show more statistics related to the plot that was clicked on.

The dialog box provides the user with two features to better help visualize traffic data: attribute filtering, and view-type specification. Attribute filtering allows the user to filter out unwanted data according to the attributes of the flows or packets. This allows users to visualize only data related to the selected metrics and analysis method. Filtering is accomplished first by choosing the visualization type. The available options are: link view, IP view, and protocol view. After the visualization type selection, users can further filter data down by selecting various options or setting parameters located in the dialog box tabs. View-type specification allows the user to specify which plots or views he or she desires to visualize. Available views include line plots, bar plots, histograms, connection views, etc. Figure 10 shows some examples of user-defined data analysis and graphical views. Among the charts, the top-left chart shows the change of entropy on the link, which is calculated using the algorithm introduced in the paper [7]. On the uplink of a worm-scanning host, we can choose entropy chart on destination IP to verify the dispersiveness of destination addresses. In a DDoS experiment, an entropy chart on source IP may be more suitable to disclose the distributed characteristics of the attack. Users can modify the length of the entropy window and other parameters to test the sensitiveness of their algorithms on the experimental data.

6. SUMMARY

In this article we presented an integrated experiment specification and visualization toolkit that supports network experimenters to conduct interactive experiments on the network testbeds such as DETER and Emulab. The ESVT package includes a topology builder including experiment specification, a TCL script generator, and various visualization tools. The freely distributed toolkit has been used in a number of network security experiments and new functions are constantly being incorporated. Besides the functionalities we have introduced in this article, the ESVT supports visualization of testbed machine status such as CPU utilization and memory usage, and visualization of BGP events and routing tables from a BGP simulator which is being developed at PSU.

Graphical user interface and visualization are highly useful means to support network experiments and security studies. Understanding of experimental needs is important to further improve the design of supporting tools in the future.

7. REFERENCES

[1] R. Bearavolu, K. Lakkaraju, W. Yurcik, H. Raje, A Visualization tool for Situational Awareness of Tactical and Stragetic Security Events on Large and Complex Computer Network, Milcom, 2003.

[2] T. Benzel, B. Braden, D. Kim, C. Neuman, A. Joseph, K. Sklower, R. Ostrenga, and S. Schwab. Experience with DETER: A Testbed for Security Research. in the

2nd IEEE Conference on testbeds and Research Infrastructures for the Development of Networks and Communities, Spain, 2006.

[3] http://www.emulab.net

[4] http://emist.ist.psu.edu

[5] http://www.isi.edu/nsnam/

[6] http://www.tcpdump.org/

[7] L. Feinstein, D. Schnackenberg, R. Balupari, and D. Kindred. Statistical Approaches to DDoS Attack Detection and Response. In *Proc. DARPA Information Survivability Conference and Exposition*, 2003.

[8] H. Kim, I. Kang, S. Bahk. Real-time visualization of network attacks on high-speed links. IEEE Network, Vol 18, Issue 5, pages 30-39, 2004.

[9] S. Krasser, G. Conti, J. Grizzard, J. Gribshaw, H. Owen. Real-time and Forensic Network Data Analysis Using Animated and Coordinated Visualization. In Proceedings of 2005 IEEE Workshop on Information Assuarnce and Security, West Point, NY, 2005.

[10] L. Li, S. Jiwasurat, P. Liu, G. Kesidis. Emulation of Single packet UDP scanning worms in enterprise networks. ITC'19, Beijing, China, 2005.

[11] S. Teoh, K. Ma, and S. Felix Wu. Visual-based Anomaly Detection for BGP Origin AS Change Events. In Proceedings of the 14th IFIP/IEEE Workshop on Distributed Systems: Operations and Management, October, 2003.

[12] Tmeas and Scripts, avaiable at http://www.cs.purdue.edu/homes/fahmy/software/emist/index.html.

[13] T. Takada, H. Koike. MieLog: A Highly Interactive Visual Log Browser Using Information Visualization and Statistical Analysis. In Proceedings of LISA 2002, 16th System Administration Conference, Philadelphia, 2002.

[14] N. Weaver, I. Hamadeh, G. Kesidis and V. Paxson. Preliminary results using scale-down using scale-down to explore worm dynamics. In *Proc. ACM WORM, Washington DC*, October 2004.

APPENDIX

A. GENERATED TCL SCRIPT

```
source tb_compat.tcl
set ns [new Simulator]
# Computer nodes Finished here
set n(899) [$ns node]
set n(900) [$ns node]
set n(902) [$ns node ]
set n(903) [$ns node ]
set n(906) [$ns node ]
set n(908) [$ns node ]
set n(910) [$ns node ]
set n(911) [$ns node ]
set n(913) [$ns node ]
set n(914) [$ns node ]
set n(916) [$ns node ]
....
set n(955) [$ns node]
#tb-set-ip $n(955) 10.1.99.2
set n(971) [$ns node]
# Switch nodes Finished here
set n(974) [$ns node]
set n(975) [$ns node ]
set n(976) [$ns node ]
set n(977) [$ns node ]
set n(978) [$ns node ]
set n(979) [$ns node ]
set n(980) [$ns node ]
set n(981) [$ns node ]
set n(982) [$ns node]
# Router nodes Finished here
set n(983) [$ns node]
# Internet/Network nodes Finished here
set lan0 [$ns make-lan "$n(899) $n(902)"
100Mb 0ms]
#Total Switch:0, Computer:20, Susceptible ones:12.
set lan1 [$ns make-lan "$n(900) $n(902)"
100Mb 0ms]
#Total Switch:0, Computer:20, Susceptible ones:10.
set lan3 [$ns make-lan "$n(902) $n(900) $n(906)
$n(899) $n(974)" 100Mb 0ms]
set lan7 [$ns make-lan "$n(906) $n(902)"
100Mb 0ms]
#Total Switch:0, Computer:20, Susceptible ones:7.
set lan68 [$ns make-lan "$n(967) $n(964) "
100Mb 0ms]
#Total Switch:0, Computer:17, Susceptible ones:8.
set link969 [$ns duplex-link $n(983) $n(981)
100Mb 0ms DropTail]
tb-set-ip $n(983) 10.99.3.2
# Running programs section
tb-set-node-startcmd $n(899) "/run_virtual
n-899-lan0 600"
tb-set-node-startcmd $n(900) "/run_virtual
n-900-lan1 600"
tb-set-node-startcmd $n(911) "/run_virtual
n-911-lan12 600"
tb-set-node-startcmd $n(913) "/run_tcp 913 600"
tb-set-node-startcmd $n(916) "/run_virtual
n-916-lan17 600"
tb-set-node-startcmd $n(921) "/run_virtual
n-921-lan22 600"
tb-set-node-startcmd $n(971) "/run_virtual
n-971-lan72 600"
tb-set-node-startcmd $n(974) "/run_tcp 974 600"
tb-set-node-startcmd $n(982) "/run_tcp 982 600"
tb-set-node-startcmd $n(983) "/run_internet
983 600"
$ns rtproto Static
$ns run
```