Scalable Network Traffic Visualization Using Compressed Graphs

Lei Shi*, Qi Liao†, Xiaohua Sun‡, Yarui Chen§ and Chuang Lin§

***Abstract*—The visualization of complex network traffic involving a large number of communication devices is a common yet challenging task. Traditional layout methods create the network graph with overwhelming visual clutter, which hinders the network understanding and traffic analysis tasks. The existing graph simplification algorithms (e.g. community-based clustering) can effectively reduce the visual complexity, but lead to less meaningful traffic representations. In this paper, we introduce a new method to the traffic monitoring and anomaly analysis of large networks, namely Structural Equivalence Grouping (SEG). Based on the intrinsic nature of the computer network traffic, SEG condenses the graph by more than 20 times while preserving the critical connectivity information. Computationally, SEG has a linear time complexity and supports undirected, directed and weighted traffic graphs up to a million nodes. We have built a Network Security and Anomaly Visualization (NSAV) tool based on SEG and conducted case studies in several real-world scenarios to show the effectiveness of our technique.**

***Keywords*-Security; Visualization; Graph Compression;**

I. Introduction

There has been a recent surge of the network traffic in both the Internet domain and the local area network, such as enterprise private networks and data center networks. Measuring and analyzing these traffic is made convenient lately through the development of Software-Defined Networking (SDN) and protocols such as OpenFlow [1]. In the process of dealing with the network traffic, the visualization of overall connection patterns, assumably as node-link graphs, is vital in many scenarios. For example, in a company's virtual private network, the administrators need a way to get access to the latest traffic patterns to maintain situational awareness. Upon security alerts, they need to conduct interactive traffic analysis to issue responsive mitigation and relevant troubleshooting.

However, it is challenging to visualize the network traffic graph scaling to a large number of hosts and complex connection patterns. First, the quadratic-complexity force-directed drawing methods for the general graph [2] can not calculate a good layout in real time ($\sim$1s) for a graph with more than a hundred nodes. Meanwhile, the number of hosts in an Ethernet can easily reach multi-thousands without counting different ports. Second, even if faster large graph layouts can be computed in servers through optimizations [3] [4], the visual clutter in the node-link representation (as in Figure 1(b)), mainly the edge crossings, prohibits the user from understanding the network traffic in details. Third, the graph clustering algorithms with cohesive or distance based measures, also known as community detection algorithms, can greatly reduce the visual complexity by the multi-scale graph abstraction approaches [5] [6]. Though quite successful in analyzing social networks, these methods can lead to poor results when communities are not prevalent, which is the case for most network traffic graphs. An visual example is shown in Figure 1(c). Moreover, the clustered top view hides the context and topology details, which are critical to traffic pattern analysis and discovery.

In this paper, we introduce a new graph simplification method to the network traffic visualization problem, based on the concept of structural equivalence [7] [8] well-known in the social network research field. Rather than detecting communities, structural equivalence classifies the network nodes into categories by their position taken in the network, depending merely on the network graph. Empirically this can be better than the community-based clustering methods in both reducing the graph size and preserving the connectivity information, due to the high-frequency subgraph patterns in the network traffic, such as singletons, hubs, and connectors (Figure 1(b)). As shown in Figure 1(a), a 3460-host traffic graph is reduced to 18 grouped nodes and 28 edges, while the same graph by the modularity clustering [9] generates 50 intermediate cluster nodes (a maximal depth of 4), 939 unclustered nodes and 15438 edges in the top view. The similar node pairing or grouping ideas have previously been studied in [10] [11], as summarized in Section II, but none of them develop and explore the method in an interactive network traffic visualization context.

In more detail, we propose our main algorithm, Structural Equivalence Grouping (SEG), in Section III. SEG completes in linear time for sparse traffic graphs. Beyond the undirected graphs, we extend SEG to support the directed and weighted graphs. We also develop the fuzzy SEG method to control the visual complexity in a finer granularity. The visual

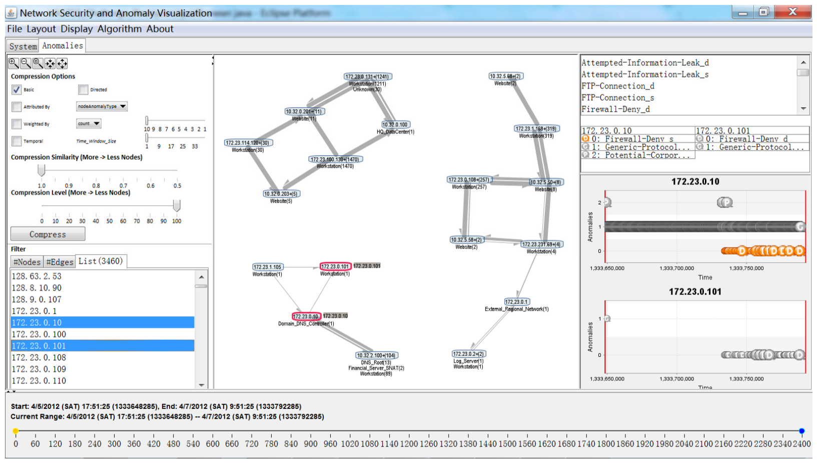
* State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Email: shil@ios.ac.cn

† Department of Computer Science, Central Michigan University, Email: qi.liao@cmich.edu

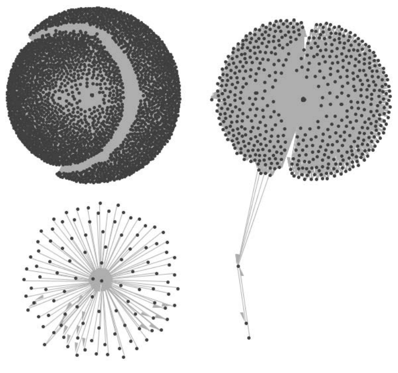
‡ College of Design and Innovation, Tongji University, Email: xsun@tongji.edu.cn

§ Department of Computer Science and Technology, Tsinghua University, Email: chenyarui@tsinghua.org.cn, chlin@tsinghua.edu.cn

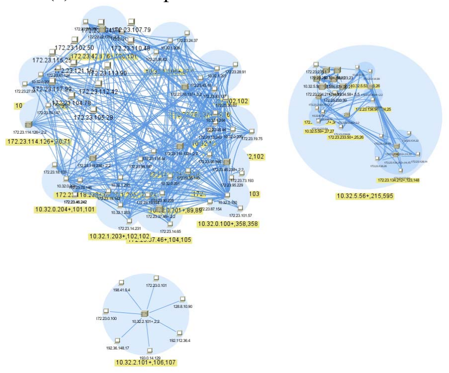
This work is supported in part by the National Basic Research Program of China (973 Program) under Grant No. 2010CB328105, National Science Foundation of China (NSFC) under Grant No. 60932003 and 60973144.


Network Security and Anomaly Visualization
File Layout Display Algorithm About
System Anomalies
Compression Options
Compression Similarity (More -> Less Nodes)
Compression Level (More -> Less Nodes)
Compress
Filter
172.23.0.10
172.23.0.101
Start: 4/5/2012 (SAT) 17:51:25 (1333648285), End: 4/7/2012 (SAT) 9:51:25 (1333792285)
Current Range: 4/5/2012 (SAT) 17:51:25 (1333648285) -- 4/7/2012 (SAT) 9:51:25 (1333792285)

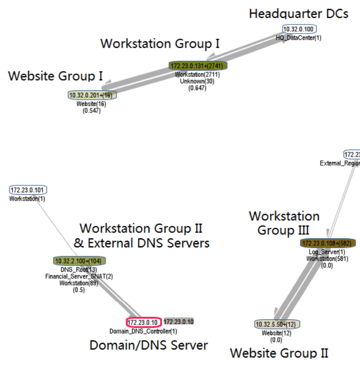
(a) SEG-compressed view in NSAV tool



(b) Original view



(c) Clustered view



(d) Manual grouped view after SEG

Figure 1. VAST 2012 Mini Challenge-II network traffic graph in 40 hours (3460 nodes, 48599 edges), under different visualization approaches.

encodings and interaction methods for the SEG-compressed traffic graph are introduced in Section IV. Notably we support both SEG-defined and manual node grouping for flexible graph navigation and analysis, as shown in Figure 1(d). The proposed SEG-based network traffic visualization method is integrated into a tool called Network Security and Anomaly Visualization (NSAV), which is evaluated in the case studies under the computer network and security scenarios in Section V. Results show that the NSAV tool can significantly improve the domain user's capability in their traffic graph understanding, analysis and detail-accessing tasks.

II. Related Work

A. Structural Equivalence

The theory of structural equivalence dates back several decades to the seminar work by social scientists Lorrain and White [7]. In this very first paper, a categorical approach for the algebraic analysis of social networks is proposed. The aim is to reduce the social structure for better empirical study on individuals' relationships. The structural equivalence is defined between two individuals who have the same type of relationships with any others in the network.

Though the concept of structural equivalence is very neat and easy to implement, in real world very few individuals in the network share exactly the same relationships. Many relaxing definitions of structural equivalence are proposed later. In [12], automorphic equivalence is developed that the two individuals are equivalence if they are swappable together with some related others while keeping all the network relationships intact. To better capture the notion of social roles, the concept of structural relatedness [13] and later regular equivalence [8] are proposed. Rather than requiring exactly the same relationships with any others, two individuals are equivalent if they share the same set of neighborhood types.

B. Graph Visualization by Node Grouping

The most relevant visualization work is the graph drawing with modular decomposition [10]. Rooted in the graph theory, the module of a graph defines a set of nodes that all nodes in the set are either neighbors or non-neighbors simultaneously to all the other nodes outside the set. Meanwhile, the modular decomposition is the process to recursively partition the graph into a tree where every tree node is a module of its parent. In [10], an algorithm to draw a graph bottom-up along its modular decomposition tree is proposed. The results are shown to reveal the graph structure while preserving several layout aesthetics. The similar work of motif simplification is done in [11] where the frequent local structures in graph, such as fans, connectors and cliques, are extracted and rendered as common motif

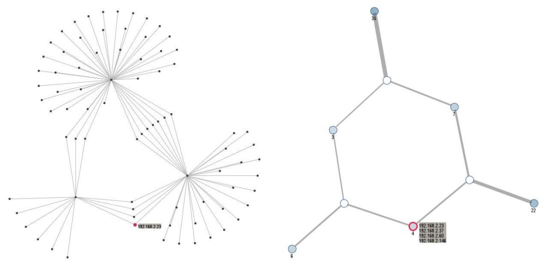

(a) Original graph (b) Compressed graph

Figure 2. SEG on an undirected graph.

glyphs encoding the type, size and specification. In the large graph drawing context, the property of neighborhood set is applied in an iterative coarsening process to reduce the graph to smaller ones and layout recursively by the multi-level drawing approach [14] [4]. The latest work in [15] applies the node grouping to the representation of graph adjacency matrices for visualizing gene regulatory networks. For a comprehensive survey in the area of large graph visualization, readers can refer to the paper in [16].

III. Algorithm

A. Structural Equivalence Grouping

Intuitively, SEG aggregates the graph nodes with the same neighbor set together into groups and construct a new graph for visualization. For example in the traffic graph of Figure 2(a), the host "192.168.2.23" can be combined with the other three surrounding hosts with exactly the same connection pattern. The new graph after SEG (Figure 2(b)) is called the *compressed graph*. The compressed graph has two kinds of nodes: the *single-node* remaining from the original graph (drawn in hollow) and the *mega-node* grouped from multiple *sub-nodes* in the original graph (drawn with filled colors). Before we formally describe the algorithms, graph terminologies throughout this paper are first defined.

Definition. Let $G = (V,E)$ be a directed, weighted and connected original graph where $V = \{v_1,...,v_n\}$ and $E = \{e_1,...,e_m\}$ denote the node and link set. Let W be the graph adjacency matrix where $w_{ij} > 0$ indicates a link from v_i to v_j, with w_{ij} denoting the link weight. In each row of W, $R_i = \{w_{i1},...,w_{in}\}$ denotes the row vector for node v_i, representing its connection pattern. The compressed graph after SEG is denoted as $G^* = (V^*,E^*)$. The compression rate is defined by $\Gamma = 1-|V^*|/|V|$ ($1-|E^*|/|E|$ in Section III-D).

The basic SEG algorithm takes the graph as a simple, undirected and unweighted one by setting $w_{ii} = 0$ and $w_{ij} = w_{ji} = 1$ for any $w_{ij} > 0$.

Structural Equivalence Grouping. On graph G, for any collection of nodes with the same row vector (including the single outstanding node), aggregate them into a new mega-node/single-node $Gv_i = \{v_{i_1},...,v_{i_k}\}$. All Gv_i form the node set V^* for the compressed graph G^*. Let $fv_i = v_{i_1}$ denote the first sub-node in Gv_i. The link set E^* in G^* are generated by replacing all fv_i with Gv_i in the original link set, and removing all the links not incident to any fv_i. SEG is single-pass on any graph, as any two nodes in the compressed graph have different row vectors.

Directed Graph. The adjacency matrix W is transformed to encode the both connection directions for each node. Each row vector $R_i (i = 1,...,n)$ becomes $R_i = \{w_{i(-n)},...,w_{i(-1)},w_{i1},...,w_{in}\}$ having $w_{i(-j)} = w_{ji}$ for $j = 1,...,n$.

Weighted Graph. The adjacency matrix W is switched to the weighted one by mapping a numeric data attribute of link (i, j) (e.g. flow count in the traffic graph) to w_{ij} in the matrix. To further increase the compression rate, discretization of the link weight is allowed: transform all link weights into $w_{ij} \in (0,1]$ by either linear or non-linear normalization, and then pick a bin count $B(B \geq 1)$ and regenerate link weights by $w_{ij} = \lceil w_{ij} \times B \rceil$.

Supporting Clique. By the basic SEG, the sub-nodes within a mega-node do not have any intra-group link. However, it is also useful to group a clique of nodes with the same external connection together. Specific rendering can be applied to differentiate between mega-nodes with isolated and fully connected sub-nodes. We devise a two-step approach to achieve that: in the first step, the graph adjacency matrix W is set to $w_{ii} = 0$, allowing the grouping of isolated nodes. In the second step, W is reset to $w_{ii} = 1$ and all the original nodes not aggregated in the first step are grouped again.

B. Controlling the Compression Rate

SEG is a deterministic algorithm in that for the same original graph, it always produces the same compressed graph. In the real usage, the user would like to flexibly control the visual complexity after the compression.

Fuzzy Structural Equivalence Grouping. The basic idea of fuzzy SEG is to group nodes with not only the same, but also the similar neighbor set. The compression rate can be increased with bounded compensation on accuracy. The key is to define the pairwise similarity score between graph nodes. Here we adopt the standard Jaccard similarity between two sample sets A and B by $J(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|}$. For the directed and weighted graphs, we introduce a unified Jaccard similarity computation between node v_i and v_j in graph G by $\rho = \frac{\sum_{\forall k} \min(w_{ik},w_{jk})}{\sum_{\forall k} \max(w_{ik},w_{jk})}$. Note that for the directed graph, $k = -n,...,-1,1,...,n$. Fuzzy SEG is achieved by setting a similarity threshold ξ. The pair of nodes with $\rho \geq \xi$ are grouped together iteratively.

C. Implementation

Structural Equivalence Grouping. The core step of SEG to group nodes with the same row vector is achieved through an appropriate hash function $H(R_i)$ over the row vector identifiers. The row vector identifier is created by splicing the positive cells in the row into a string. This hash-based implementation has a computational complexity of $O(ND)$.

Table I
SEG PERFORMANCE ON VAST11 CHALLENGE DATASET.

Data	edges (before)	edges (after)	**rate (edges)**	**time (SEG)**	layout (before)	**layout (after)**
undirected, sim=1	1613	50	**96.9%**	**0.007**	0.242	**0.084**
undirected, sim=0.8	1613	39	**97.6%**	**0.012**	0.242	**0.088**
undirected, sim=0.5	1613	23	**98.6%**	**0.006**	0.242	**0.079**
directed, sim=1	1613	82	**94.9%**	**0.005**	0.242	**0.084**

Table II
SEG PERFORMANCE ON VAST12 CHALLENGE DATASET.

Data	edges (before)	edges (after)	**rate (edges)**	**time (SEG)**	layout (before)	**layout (after)**
undirected, sim=1	48599	28	**99.9%**	**0.437**	3.151	**0.078**
undirected, sim=0.8	48599	19	**99.9%**	**0.515**	3.151	**0.062**
directed, sim=1	48599	1022	**97.9%**	**0.328**	3.151	**0.125**
directed, sim=0.8	48599	403	**99.2%**	**0.374**	3.151	**0.078**

Table III
SEG PERFORMANCE ON TWITTER SOCIAL NETWORKS.

Data	edges (before)	edges (after)	rate (edges)	**time (SEG)**
mention undirected	122976	66858	45.6%	**2.074**
mention, sim=0.5, anchor	122976	63089	48.7%	**370.121**
mention, sim=0.5, shingle	122976	59873	51.3%	**24.92**
follower undirected	2926986	2574823	12%	**32.835**

Here N is the number of nodes in the original graph, D is the average node degree for the complexity to splice and hash each row vector identifier. Therefore, the basic SEG has a linear complexity and will perform well even for very large graphs.

Fuzzy Structural Equivalence Grouping. We introduce an improvement by the shingle-ordering [17] on the fuzzy SEG implementation. For each row vector R_i, construct the element set $A_i = \{a | w_{ia} = 1\}$. Given any permutation $\sigma : \{1, ..., n\} \rightarrow \{1, ..., n\}$, the shingle of the row vector R_i is defined by $M_\sigma(A_i) = \sigma^{-1}(min_{\alpha \in A_i}\{\sigma(\alpha)\})$. By shingle properties [17], the probability that shingles of set A and B are identical is precisely their Jaccard coefficient $J(A, B)$. The corresponding fuzzy SEG algorithm still works in a greedy manner to scan each node in a order. Each newly scanned node without a group is selected as an anchor node and also create a new group around the anchor node together with the other nodes similar enough. Using shingle-based method, we do not need to compare pairwise similarity. Instead, we pre-compute and record k shingles for each node's neighbor set, using k orthogonal permutation functions. At most kN lists are created, each holding the nodes with a same shingle result $(1 \sim N)$ by one permutation function. Then for each anchor node, we can get the set of similar enough nodes by scanning the k corresponding list, according to the shingle property. The overall complexity is $O(kND + kN^*D^2)$, where kND is the complexity to pre-compute shingles, D^2 is the average size of the 2-hop neighbor set of each node, also the upper bound of the average length of each list to scan. In most graphs, the complexity is significantly below N^2.

D. Performance

We evaluate the SEG performance in terms of the visual compression rate (by the number of edges), the compression time and the layout time before and after SEG. All the experiments are carried out on the same 64-bit Windows desktop (Intel Core i7@3.40GHz with 8GB memory). Table I and II show the results on two traffic graph datasets.

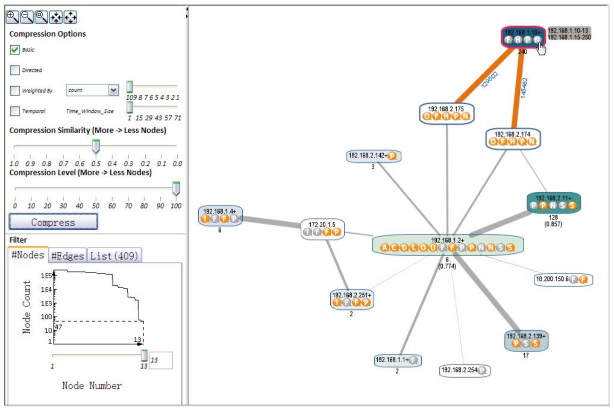


Figure 3. User interface for compressed graph visualization. Left: SEG controllers. Right: main panel for traffic visualization. Data set is from VAST Challenge 2011.

Notably for both cases, SEG achieves a more than 95% compression rate with the basic algorithm or applying a fuzzy setting. The compression time is mostly below 0.5 second, reaching up to 10^4 edges. The layout time after SEG is significantly reduced from that of the original graph.

We also experiment on one type of extreme-scale graph: the twitter social graphs with up to millions of edges (Table III). Because these graphs are beyond the application domain of SEG, we only report their running time for the purpose of a scalability study. The basic SEG compresses the largest graph in half a minute; the fuzzy SEG with the shingle implementation supports graphs with up to 10^5 edges and returns results in 25 seconds.

IV. VISUALIZATION

A. Compressed Graph Visual Encoding

The right panel of Figure 3 gives an example of the SEG-compressed graph visualization. As shown in the figure, the mega-nodes are differentiated from the single-nodes by the fill color. The single-nodes have no fill and the mega-nodes have standard fill colors, with the color saturation mapped to the number of sub-nodes in the group. The larger group is filled with the more saturated color. By default, the fill color hue is blue, e.g. the top-right node "192.168.1.10+". For the mega-nodes created by the fuzzy SEG, e.g. the node "192.168.2.11+" on the right, the fill color will gradually shift from blue to green and then to brown, according to the smallest pairwise similarity score within the group. The mega-node containing totally dissimilar nodes will lead to a pure brown fill color. We do not use the node size to represent the group size of the mega-node, since the group size normally has a rather biased distribution. The large groups can introduce unnecessary visual complexities which counters our initial design goal.

Labels of the mega-node are created by aggregating the labels of the sub-nodes in the original graph. Due to the space limitation, an abstracted label is drawn on each mega-node as the node identifier. The full label will pop-up upon a mouse hovering or a click action, e.g. the one on the node “192.168.1.10+”. The group size of each mega-node is drawn below the visual node, together with the intra-group similarity score when it is below one. By default, straight lines are used to represent the links in the compressed graph, with thickness mapped to the sum of counts of all the corresponding links in the original graph. Compressed graph attributes can also be mapped flexibly into the visuals. For example, the node label can be the host types (Figure 1(a)) or the alphabetical anomaly icons (Figure 3) indicating the type of anomalies happening on the host.

B. Graph Interaction Design

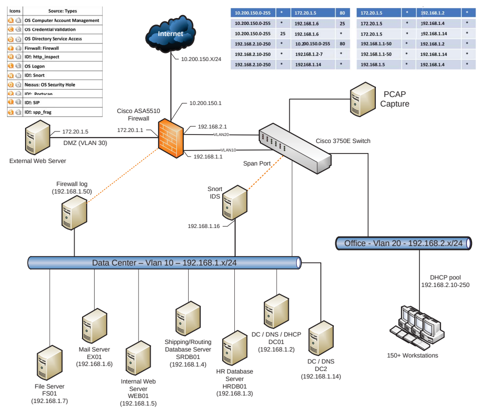
Except the basic graph interactions, more controls over the SEG setting are accessible through a control panel as in the top left of Figure 3. In the “Compression Options” section, multiple checkboxes work as switches for the basic, directed and weighted SEG. For the weighted SEG, the link weight mapping from the graph attribute can be specified. For normalized link weight, a bin number can be selected to discretize the weight. In the “Compression Level” and “Compression Similarity” sections, a larger or smaller compressed graph is tunable by the LOD control and the fuzzy SEG.

Complementary to the automatic SEG operation, we also introduce the manual node grouping/splitting interaction as in many cases the users have their own criteria towards a best traffic abstraction. The user can either select a collection of nodes and click the “group” button in the navigation panel, or use drag-and-drop to group one node into another once per time. In the drag-and-drop process, the pairwise similarities with all the other nodes are shown as visual hints. In a manual splitting process, the user can either select some mega-nodes and click the “split” button, or just double-click one mega-node. The mega-node grouped by the fuzzy SEG will collapse to mega-nodes by the basic SEG, and further collapses to the original sub-nodes.

C. Network Security and Anomaly Visualization Tool

In the integrated NSAV tool (Figure 1(a)), the compressed traffic graph visualization functions as the major view in the central panel. Meanwhile, there are several other panels illustrating other facets of the network traffic.

Graph Node/Edge Filtering and Selection. This is in the lower-left corner of the control panel in Figure 3. In the “node” tab, the user can filter the graph according to various criteria of the host importance. A host importance distribution is shown on top of the slider to suggest a better filter setting. In the “edge” tab, the graph is filtered according to the connection importance respectively, The mapping of the node/edge filtering criteria can be manually adjusted according to the available attributes. Finally, in the “List” panel, any individual host can be quickly selected from a list so that the host will be highlighted in the traffic graph.



Icons	Source: Types
A A	OS Computer Account Management
C C	OS Credential Validation
D D	OS Directory Service Access
F F	Firewall: Firewall
H H	IDS: http_inspect
L L	OS Logon
N N	IDS: Snort
O O	Nessus: OS Security Hole
P P	IDS: Portscan
I I	IDS: SIP
S S	IDS: spp_frag

10.200.150.0-255	*	172.20.1.5	80
10.200.150.0-255	*	192.168.1.6	25
10.200.150.0-255	25	192.168.1.6	*
192.168.2.10-250	*	10.200.150.0-255	80
192.168.2.10-250	*	192.168.1.2-7	*
192.168.2.10-250	*	192.168.1.14	*

172.20.1.5	*	192.168.1.2	*
172.20.1.5	*	192.168.1.4	*
172.20.1.5	*	192.168.1.14	*
192.168.1.1-50	*	192.168.1.2	*
192.168.1.1-50	*	192.168.1.14	*
192.168.1.5	*	192.168.1.4	*

Internet
10.200.150.X/24
10.200.150.1
Cisco ASA5510 Firewall
172.20.1.5
172.20.1.1
DMZ (VLAN 30)
External Web Server
192.168.2.1
VLAN20
VLAN10
192.168.1.1
PCAP Capture
Cisco 3750E Switch
Span Port
Firewall log (192.168.1.50)
Snort IDS
192.168.1.16
Office - Vlan 20 - 192.168.2.x/24
Data Center – Vlan 10 – 192.168.1.x/24
DHCP pool 192.168.2.10-250
150+ Workstations
File Server FS01 (192.168.1.7)
Mail Server EX01 (192.168.1.6)
Internal Web Server WEB01 (192.168.1.5)
Shipping/Routing Database Server SRDB01 (192.168.1.4)
HR Database Server HRDB01 (192.168.1.3)
DC / DNS / DHCP DC01 (192.168.1.2)
DC / DNS DC2 (192.168.1.14)

Figure 4. AFC network topology. Top-left: anomaly icon for event types. Top-right: acceptable flow rules.

Anomaly Panel. The right panel of the NSAV tool is referred as the anomaly panel (Figure 1(a)), which consists of three sections. The first (top) panel includes a list of all the anomaly types occurred during the specified time range. Selection of any anomaly types from the list will highlight in the traffic graph all the hosts on which such anomalies have happened. The second (middle) panel consists of mappings between anomaly icons and their anomaly type description, much like a legend for the anomaly timeline visualization below. The third (bottom) panel shows anomaly events as timeline plots of all the interesting hosts under investigation. Each unique anomaly event at a specific time will have an anomaly icon to encode its type and count (orange color represents the source and gray color represents the destination for an anomaly connection).

Time Range Selector. The double-end slider at the bottom of the tool allows the user to interactively select the desired time range for the investigation period.

V. Case Study

We first describe the method to process the network traffic data in NSAV tool and then present two case studies.

A. Network Traffic Processing

The NSAV tool reads and automatically correlates several standard network traffic and management data:

Netflow is the industry standard in network management to collect and monitor network activities. Each Netflow record contains information for a packet flow, e.g. src/dst IP addresses and port numbers, protocol, flow size, start and end timestamps, etc.

Intrusion Detection System (IDS) like Snort is a fully functional and customizable system to monitor different types of network traffic. IDS log includes all kinds of intrusion detection events.

Acceptable Use Policy (AUP) of each organization is about what activities/services are allowed within its network. AUP can be transformed into a good rule set for visual anomaly detections.

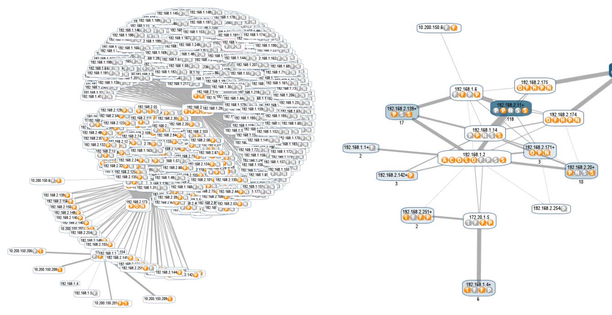

(a) Original (b) Compressed

Figure 5. AFC corporate network traffic overviews.

Operating System Log records events such as invalid logon attempts, as well as events related to resource use, such as creating, opening, and deleting files.

In detail, the NSAV tool first reads the Netflow data and builds network flow graphs on each time window (an hour by default). The flow graph of any consecutive time windows is generated on the fly by merging the per-window graphs. The network anomalies are extracted from AUP files, IDS logs, OS logs and other reports, and then processed into standard anomaly files for each host. For example, the firewall anomalies are detected by first translating the AUP into the Acceptable Flow Rules (top-right of Figure 4) and filtering the Netflow entries with the rules. The anomaly type list is given in the top-left of Figure 4.

B. *Situational Awareness*

We first apply the NSAV tool on the VAST 2011 Mini Challenge-II dataset [18]. The dataset includes a computer network architecture (Figure 4) of a shipping company - All Freight Corporation (AFC) and all the necessary traffic data for the tool, including three days of Netflow firewall log. Note that we drop the src/dst port number when aggregating the flow graphs, so as to reduce the traffic graph size.

We give a detailed user trail as below. Consider John, the AFC network operation lead, is checking the corporate network status of the recent three days for noteworthy events. He starts by loading the whole network traffic in this period, as shown in Figure 5(a). Because the view is too messy, he continues by applying the basic SEG to create a compressed traffic graph, as shown in Figure 5(b). From this graph, he quickly learns some key hosts in the period (e.g. 1.2, 1.14. "192.168." is omitted throughout this study), but still feel a little prohibitive to proceed to details. He further simplify the graph by using the fuzzy SEG with a similarity score of 0.5. The resulting graph in Figure 3 is clear enough for his overview purpose: the hosts in the central group (1.2, 1.6, 1.14, 2.171-173) and 2.174/175 are all the hub nodes.

Port Scan & OS Security Holes. Based on the AFC network structure (Figure 4), John bypasses three server machines (1.2, 1.6, 1.14) which routinely communicate with all the hosts for DNS/data services. Also in his knowledge, the suspicious behaviors of a hub node, e.g. port scan, often

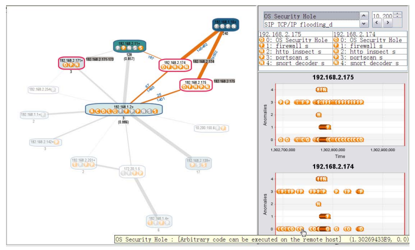

OS Security Hole
SIP TCP/IP flooding_d
192.168.2.175
192.168.2.174
Time
OS Security Hole : Arbitrary code can be executed on the remote host

Figure 6. The hosts with security holes and the cross-subnet port scans from 192.168.2.174/175.

associate with the OS security holes. So he clicks on this anomaly type and highlights all the hosts with such anomaly on the graph. To drill-down to individual hosts, he splits the fuzzy group and locates 2.171-2.175 as the threats. He finds that 2.174/175 are more dangerous due to the higher port-scan rate (by the link thickness) and the cross-subnet floods to 1.10-250 where many hosts do not exist. The screenshot with anomaly views of 2.174/175 is given in Figure 6.

DoS Attacks. A critical machine John examines in the following is the AFC's external web server (172.20.1.5). With the SEG-compressed graph, this server is easily located by its unique connection patterns. A single click on the host shows up a noteworthy anomaly icons (I) on the morning of the first day, suggesting that there could be Denial-of-Service (DoS) attacks through SIP. John further drills down to that period with the time range selector and highlight the web server's egocentric traffic graph. Figure 7 confirms the potential DoS attacks from the external hosts 10.200.150.201, 206-209, through the anomalies happening simultaneously with the web server.

C. *Botnet Detection*

We evaluate on VAST 2012 Mini Challenge-II dataset from a financial company's network [19]. An overview of the 40-hour network traffic by NSAV tool is shown in Figure 1(a). Noteworthy events are detected in a divide-and-conquer manner over each of the connected components.

IRC Malware Infection and Botnet Behavior. In the first subgraph of the network traffic, as in Figure 8, it is identified that the I and M icons appeared frequently and almost in couples in reverse directions. A selection of the $IRC-Malware-Infection\_s$ anomaly (icon I) in the anomaly type list reveals three group of hosts highlighted in red in Figure 8. They are all workstations having enormous IRC connections to a portion of 12 websites (10.32.5.*), potentially to be compromised botnet clients. Further selecting two typical workstations (172.23.123.105, 172.23.231.174) and websites (10.32.5.50, 10.32.5.52), their temporal anomaly distributions are plotted in the right panel of Figure 8. It is shown that the IRC traffic exchanged with the websites overwhelm in the whole inspected period. Note that nine of the websites (10.32.5.51-59) reply with the IRC authorization message (icon M), indicating the establishment of potential botnet server-client connections.

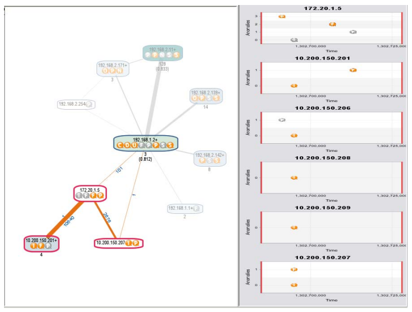
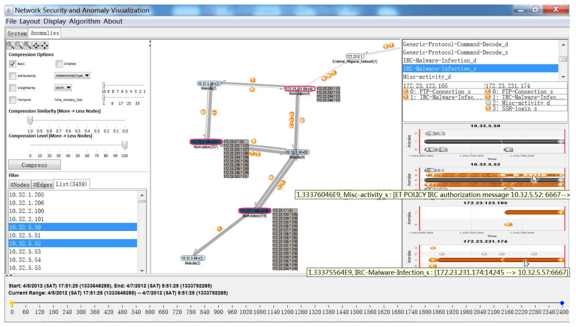


Figure 7. DoS attacks against 172.20.1.5 (corporate web server) from 10.200.150.201, 206-209.



Basic ✓ Directed
Attributed By: nodeAnomalyType
Weighted By: count
Temporal: Time_Window_Size
Compression Similarity (More -> Less Nodes)
Compression Level (More -> Less Nodes)

```
Generic-Protocol-Command-Decode_d
Generic-Protocol-Command-Decode_s
IRC-Malware-Infection_d
IRC-Malware-Infection_s
Misc-activity_d
```

```
172.23.123.105                172.23.231.174
0: FTP-Connection_s           0: FTP-Connection_s
1: IRC-Malware-Infec...       1: IRC-Malware-Infec...
                              2: Misc-activity_d
                              3: SSH-login_s
```

```
10.32.1.205
10.32.1.206
10.32.2.100
10.32.2.101
10.32.5.50
10.32.5.51
10.32.5.52
10.32.5.53
10.32.5.54
10.32.5.55
```

1.33376046E9, Misc-activity_s : [ET POLICY IRC authorization message 10.32.5.52: 6667-->

1.33375564E9, IRC-Malware-Infection_s : [172.23.231.174:14245 --> 10.32.5.57:6667]

Start: 4/5/2012 (SAT) 17:51:25 (1333648285), End: 4/7/2012 (SAT) 9:51:25 (1333792285)
Current Range: 4/5/2012 (SAT) 17:51:25 (1333648285) -- 4/7/2012 (SAT) 9:51:25 (1333792285)

Figure 8. Three group of machines in heavy IRC traffic with the websites through port 6667. Potential botnet infections.

The host 172.23.231.174 (all-time IRC client) show fine-grained patterns: the connections are composed of two temporal stages, indicated by a small gap in the middle of the anomaly panel. A drill-down analysis at this gap shows that the first stage ends-up with a large port (43325) and the second stage starts with a small port (1185). After checking the anomaly file, we conclude that the IRC traffic from the workstations are programmed, with sequentially enumerated source ports. It verifies the hypothesis that these hosts have been compromised as botnet clients.

VI. Conclusion

In this paper, we apply the Structural Equivalence Grouping (SEG) method to visually condense the large network traffic graph without sacrificing the connectivity information. It is shown that SEG can effectively reduce the scale of many real-world traffic graphs and still preserve critical features of the original graph. The classical node-link representation is introduced to visualize the SEG-compressed graph, with carefully designed visual encodings and interactions to provide the user with both ease and interactivity in visual analysis. We showcase the applications of the integrated NSAV tool in two use cases and demonstrate the advantages of the proposed compressed graph visualization technique.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[2] E. R. Gansner, Y. Koren, and S. North, "Graph drawing by stress majorization," in *Graph Drawing*, 2004.

[3] P. Gajer and S. G. Kobourov, "GRIP: Graph drawing with intelligent placement," *Journal of Graph Algorithms and Applications*, vol. 6, no. 3, pp. 203–224, 2002.

[4] Y. Hu, "Efficient and high quality force-directed graph drawing," *Mathematica Journal*, vol. 10, no. 1, pp. 37–71, 2005.

[5] D. Auber, Y. Chiricota, F. Jourdan, and G. Melancon, "Multi-scale visualization of small world networks," in *InfoVis*, 2003, pp. 75–81.

[6] J. Abello, F. van Ham, and N. Krishnan, "ASK-GraphView: A large scale graph visualization system," *IEEE Trans. Visual Comput. Graphics*, vol. 12, no. 5, pp. 669–676, 2006.

[7] F. Lorrain and H. C. White, "Structural equivalence of individuals in social networks," *The Journal of Mathematical Sociology*, vol. 1, no. 1, pp. 49–80, 1971.

[8] D. R. White and K. P. Reitz, "Graph and semigroup homomorphisms on networks of relations," *Social Networks*, vol. 5, no. 2, pp. 193–234, 1983.

[9] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, no. 6, p. 066133, Jun 2004.

[10] C. Papadopoulos and C. Voglis, "Drawing graphs using modular decomposition," *Journal of Graph Algorithms and Applications*, vol. 11, no. 2, pp. 481–511, 2007.

[11] C. Dunne and B. Shneiderman, "Motif simplification: Improving network visualization readability with fan and parallel glyphs," in *CHI*, 2013, pp. 3247–3256.

[12] S. P. Borgatti and M. G. Everett, "Notions of position in social network analysis," *Sociol. methodol.*, vol. 22, pp. 1–35, 1992.

[13] L. D. Sailer, "Structural equivalence: Meaning and definition, computation and application," *Social Networks*, vol. 1, no. 1, pp. 73–90, 1978.

[14] T. A. Davis and Y. Hu, "The university of florida sparse matrix collection," *ACM Trans. Math. Software*, vol. 38, no. 1, 2009.

[15] K. Dinkla, M. A. Westenberg, and J. J. van Wijk, "Compressed adjacency matrices: Untangling gene regulatory networks," *IEEE Trans. Visual Comput. Graphics*, vol. 18, no. 12, pp. 2457–2466, 2012.

[16] T. von Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. van Wijk, J.-D. Fekete, and D. W. Fellner, "Visual analysis of large graphs," *EuroGraphics - State of the Art Report*, pp. 37–60, 2010.

[17] F. Chierichetti, R. Kumar, and S. Lattanzi, "On compressing social networks," in *KDD*, 2009.

[18] "IEEE VAST Challenge," 2011, http://hcil.cs.umd.edu/localphp/hcil/vast11/.

[19] "IEEE VAST Challenge," 2012, http://www.vacommunity.org/VAST+Challenge+2012/.