Deliverable FI3-D6.2.4

Report on visualization of network event logs, anomalies and security policies

Marko Niiranen

Tivit Future Internet Program
(Tivit FI)

Period: 1.4.2011 – 30.4.2012

Tivit, Strategisen huippuosaamisen keskittymän tutkimusohjelma

Rahoituspäätös 1171/10, 30.12.2010, Dnro 2790/31/2010

www.futureinternet.fi

www.tivit.fi

This work was supported by TEKES as part of the Future Internet programme of TIVIT (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT).

Executive summary / Internal release

Title: Visualization of network security events

Content: Report about practical iterations and decisions made when designing and implementing a tool for visualizing network security events.

Impact: The report describes key findings how network security events can be visualized and what kind of challenges there may appear. It shows how visualizations are an effective tool for finding patterns and identifying anomalies.

Contact info: Marko Niiranen, marko.niiranen@stonesoft.com

1 Introduction

The network security can be divided into two equally important parts: 1) proactive enforcement of rules, and 2) reactive monitoring of state. Traditionally a lot of effort has been put into the first part, for example by deploying firewall and intrusion prevention devices into networks. However, when the network security is facing more advanced attacks and more complicated environment with social media sites, multimedia streams, IP phone calls and such, the monitoring part is coming more and more important.

A major part of the monitoring is a search of anomalies. They may are often a hint of something unexpected or unwanted. They can be hidden into a massive amount of data which requires a lot of effort and experience to find. Statistically meaningful anomalies are easier to find with statistical reports, but minor deviations to normal are almost impossible to see.

Visualization of network events provides a simple and easy to understand tool for finding minor anomalies from massive data sets, because human eye is trained to decode complex visual images and finding patterns in them.

Unfortunately reprioritization of tasks caused that there was not time to make research about visualization of security policies. However, it is a certainly one of the future research topics to explore.

2 Concept Design

The starting point for the research was event logs provided by Stonesoft firewall and intrusion prevention devices. There was already an efficient way to receive, store and browse events in Graphical User Interface i.e. GUI Client existing. It was quite easy to build prototypes and actual feature over that.

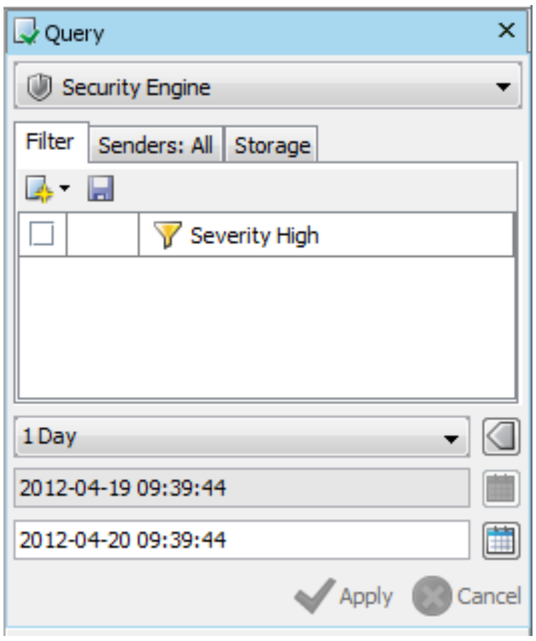
However, there was an immediate scalability problem: There can be hundreds of millions of events stored into disk, which means that it is, if not impossible, very unwise to try to visualize all of them. Therefore, it was obvious that the user needs to define a context first.

The context can be defined by three factors:

1) Time range (e.g. last 24 hours)

2) Sender device(s)

3) Filtering (e.g. events with the high severity)



Severity High

1 Day

2012-04-19 09:39:44

2012-04-20 09:39:44

Image 1. Selection of the context.

Soon we noticed that this is not enough. It is still possible to have several millions events stored for a single day. Therefore we were forced to create a hard maximum limit for amount of events which can be visualized together. After several tries our limit was set to 100 000.

It also enforced us to create a new kind of task specific view, where the selected events can be browsed, filtered and visualized.

3 Models

After a set of security events has been defined, starts the modeling work. The model describes what kind of objects and relations are visualized. In the development phase, we wanted to try as many different models as possible. This had two consequences: 1) we found out that it is convenient to have different kinds of models for different kinds of data, and 2) we noticed that there are only few really meaningful models.

3.1 Source-Destination

The first of our models consisted only source and destination IP addresses as objects and connections between them. It tries to show how clients and servers are linked together. We added also a color to edges to show the nature of connection: is they allowed or discarded by the security policy.

The color coding is:

1) Green. All connections were allowed.
2) Red. All connections were discarded.
3) Yellow. There were the both allowed and discarded connections.

Unfortunately in big environments these visualizations became soon too connected and impossible to use. Therefore it was not included to the product.

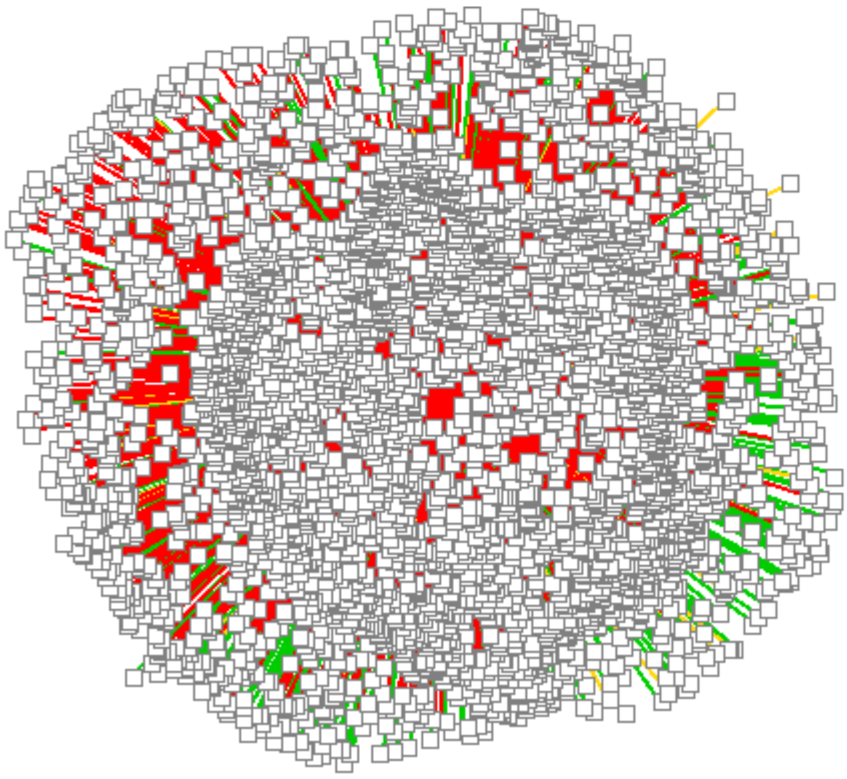


Image 2. Source-Destination visualization

3.2 Source-Service

Another of the first models was an attempt to visualize events as clients and services which are accessed. So there were two kind of nodes: clients and services, which were a combination of destination IP address and destination TCP port number. Also the type of action was added into the edge as color.

This provided a lot better results. However, the concept of service was quite hard to understand by users. Moreover, it causes that services within a single physical device, were scattered randomly over the diagram. It provided a good starting point for model research but was not that usable to be included to the final version.

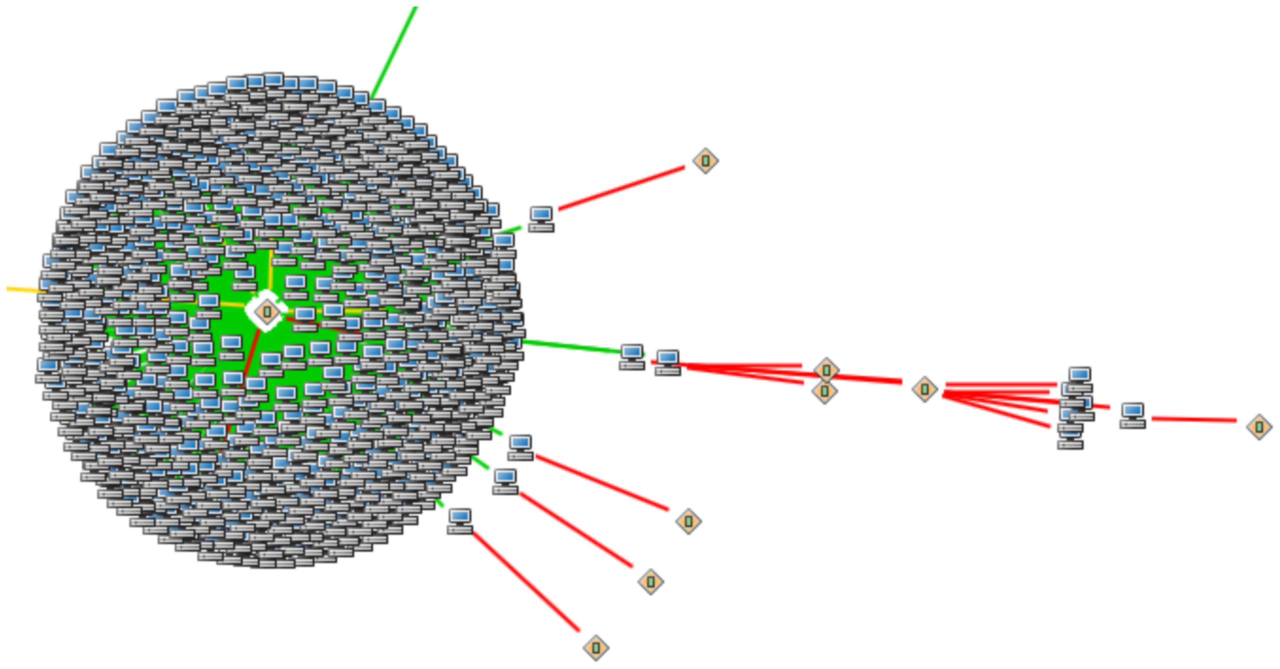


Image 3. Source-Service model.

3.3 Situation Map

The next approach was to add more situational awareness into the visualization. Therefore a new node type Situation was added. Situation identifies the kind of each event. For example, is it related to normal access control (allowed, discarded) or to deep inspection (SQL injection noticed).

Soon after adding it into the Source-Service model, it became obvious that this was not a way to go. The real time visualizations became very often extremely hard to read, especially when there were some “dominating” Situations which were very much connected.

This is why this model was abandoned.

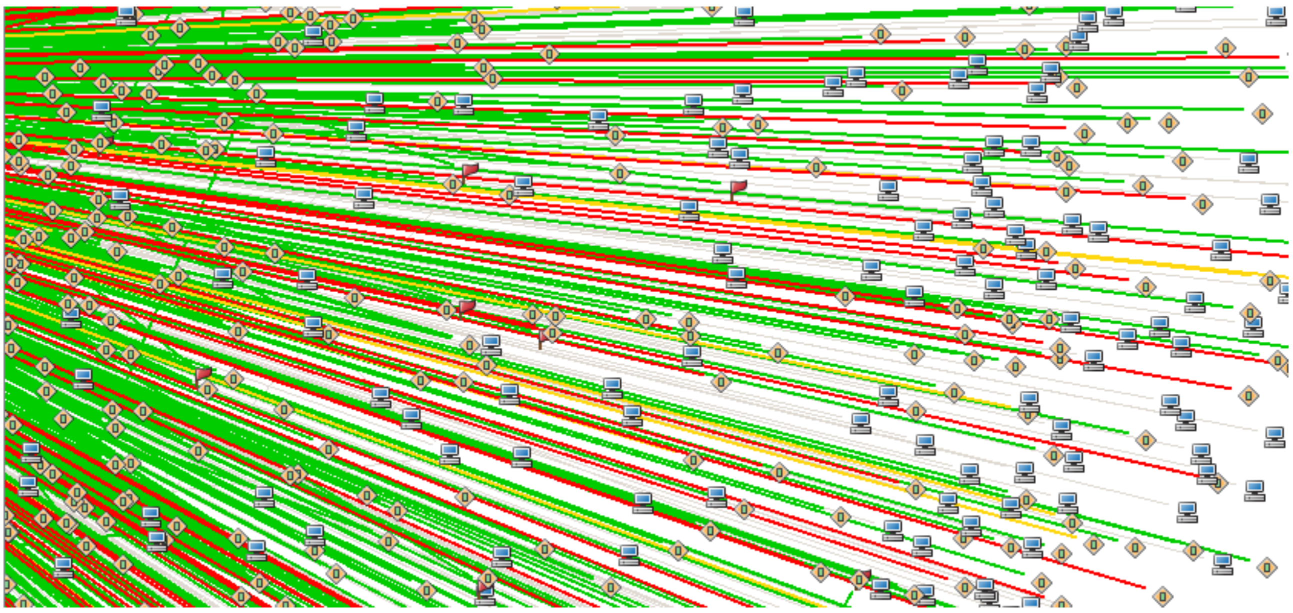


Image 4. Situation map model

3.4 Service-Situation Map

One of the most promising types of model, which was discovered, was so caller Service-Situation Map. In included four types of node: 1) source IP address, 2) destination TCP port, 3) Situation and 4) destination IP address.

This model clearly demonstrates that longer chains are better than short ones. More information is added into the visualization more beneficial it is.

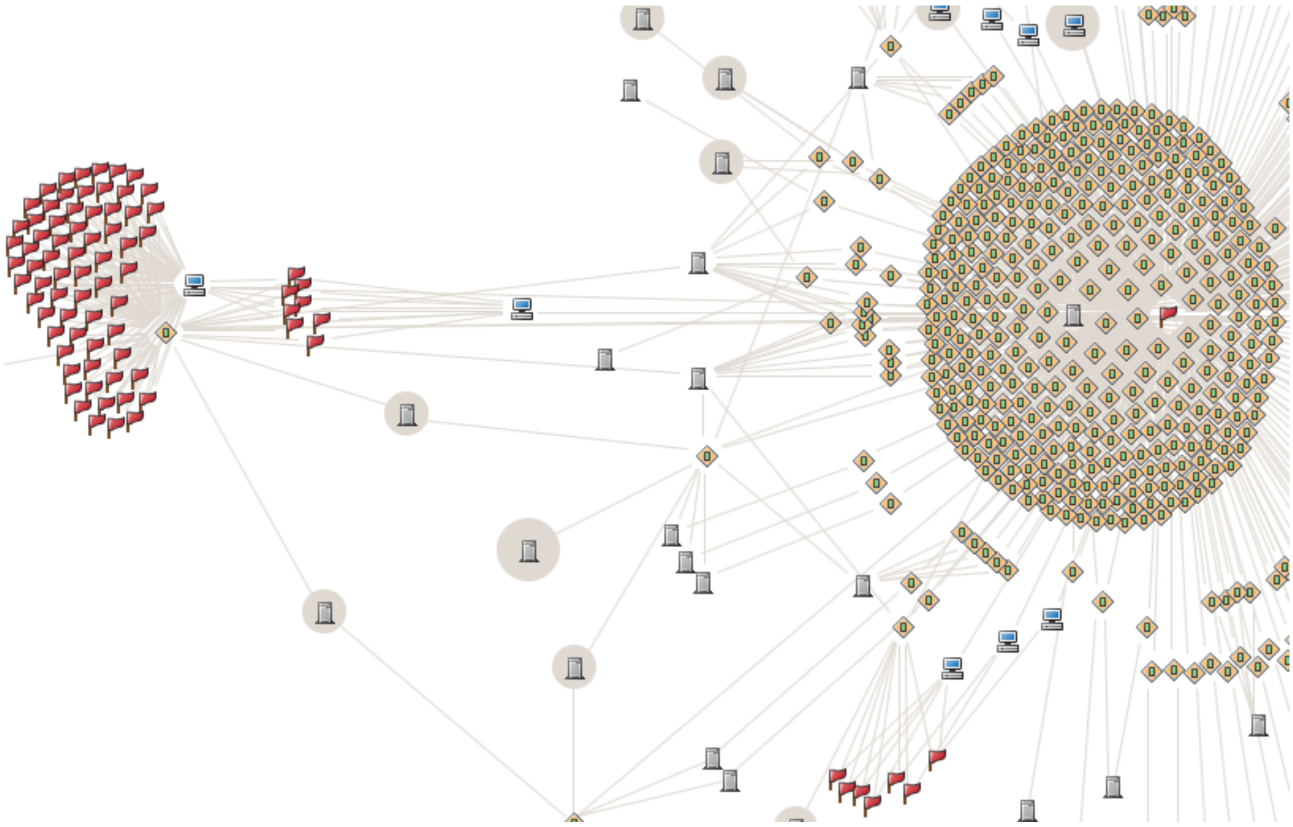


Image 5. Service-Situation Map model

3.5 Application Usage

In the modern network security, IP addresses and TCP ports are no longer the most wanted abstraction layer. There is a growing need for higher concepts and better situational awareness. Usually this means concepts like user (meaning the actual end user), and application (the content that is browsed through web browser).

Therefore also an application level variant of Service-Situation Map model was tested. It includes four node types: 1) user, 2) source IP address, 3) application and 4) destination IP address. Moreover, the action type color coding was also added to edges.

It was noticed to be so appealing that it was immediately added into the released product.

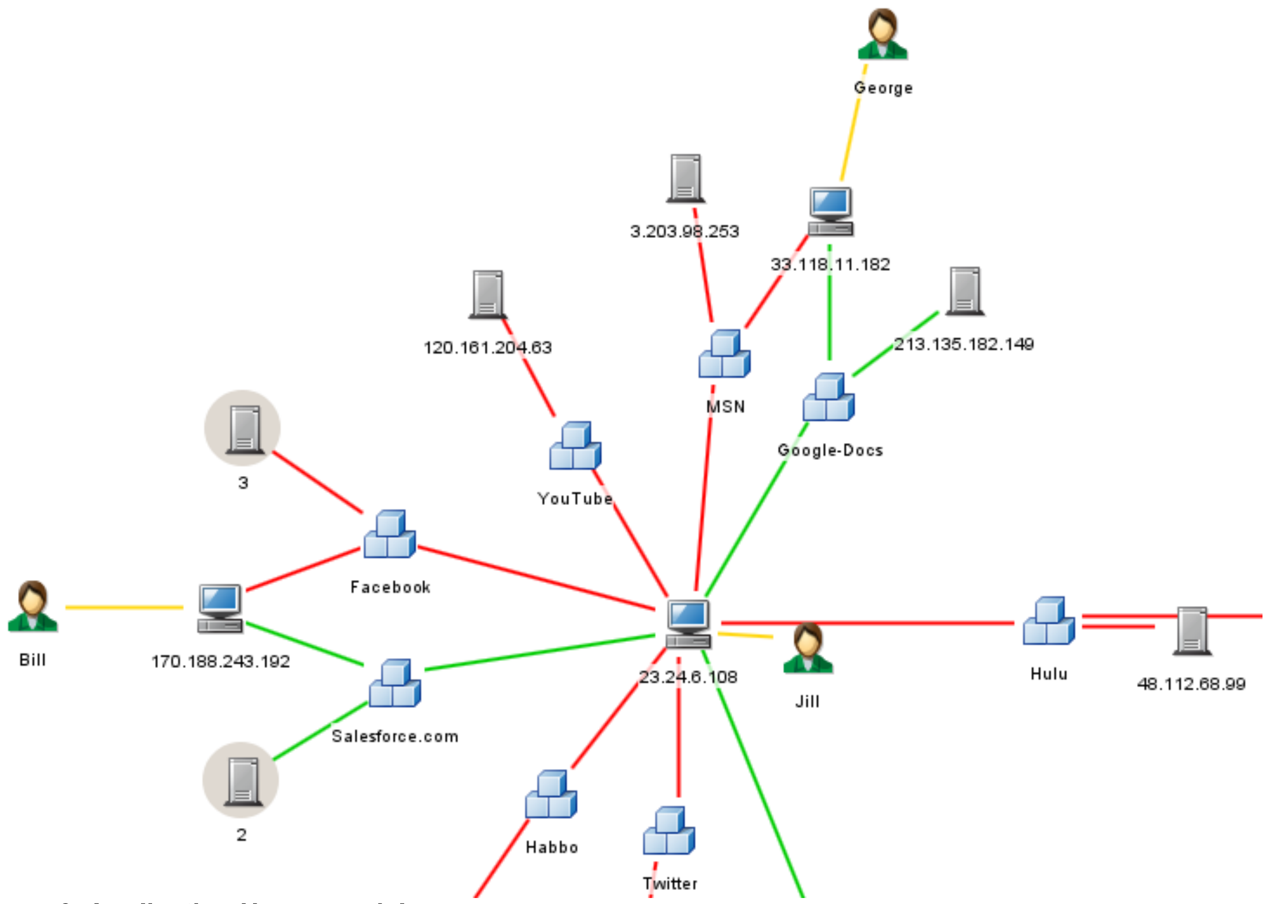


Image 6. Application Usage model

4 Outlook

The outlook of visualization is very important aspect, because it needs to provide as much information to the user as possible, but at the same time, be clear and readable. There are two basic components in the visualization: 1) nodes and 2) edges. Next, their design is explained in more detail.

4.1 Nodes

The outlook of the node was much harder task than expected beforehand. It required many iterations and tries and failures before it started to look and feel right.

Our first guideline was that the type of node should be visible. The icon was chosen to identify it. For example, a source IP address of connection is a desktop computer (client) and a destination IP address of connection is a server computer.

Another mandatory requirement was to show also the value of node as a label. It is recommended that labels are painted into the top of edges so labels don't get mixed. In addition to that, a mild fading effect was added under the label to make it more readable.

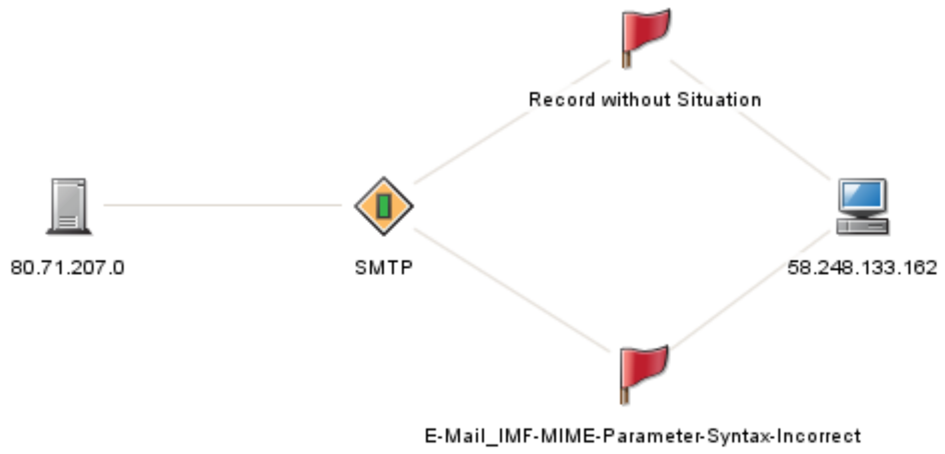


Image 7. Basic visualization of nodes.

4.2 Edges

Edges are also difficult objects to visualize. They provide core information about structure of diagram, but they should be mild enough not to be annoying. After several iterations a light gray was chosen to be the default color for them.

The encounter of edge and node is also interesting. In the first prototype, edges were drawn into nodes. However, that caused a messy impression. Then there was a circular boundary around nodes which was the end point for edges. Unfortunately that did not work either. In the finalized version there is an invisible boundary around the node where edges are ending. It gives clear and calm impression.

5 Layout

The layout is the very core of visualization tool. The layout either highlights the anomalies or hides them. Therefore it was our central focus research areas.

At the beginning, we tried to create as simple layout algorithm as possible. It tried to browse through the model and find patterns such as rows, circles and stars. However, real data is very often more complex and highly connected. That is why we were forced to abandon that approach very soon.

The second approach was to implement an algorithm that would determine the layout for the whole model at once. After some literature research, a traditional force-based algorithm was chosen. It has a three phases:

1) Initial layout. It is a semi-random algorithm which puts nodes in their initial phases
2) Raw layout. Kamada-Kawai algorithm (Kamada, 1989). It moves the most unbalanced node per iteration into the optimal place.
3) Fine layout. Fruchterman-Reingold's algorithm (Fruchterman, 1991). It moves all nodes in an oscillating manner until all the oscillation energy is used.

The parameters of algorithms were fine-tuned with a couple of real data sets. As the result, the algorithm was found to be quite robust and reasonably fast.

However, later it was noticed that the user may assume an identical visualization for every time for the same event set. This required that the totally random initial layout was changed to be semi-random. In other words, the randomization seed was fixed which guarantees that for the same data set, the same layout is always calculated.

6 Usability Enhancements

The main goal of the project was to release a real tool for users. This required an extra effort to the usability of the tool. Surprisingly many of these usability related enhancements had also an impact to the visualization itself.

In this chapter some of the most important usability enhancements and their effect to the visualization are presented.

6.1 Zooming

Zooming is one of the obvious things that are very hard to add afterwards. With large event sets the overall visualization becomes so big that it is impossible to browse it without a convenient zooming functionality.

This also adds additional requirements for the diagram canvas:

1) Coordinates must be float numbers, because integers are too rough

2) Size of the canvas should be flexible, so it can handle both small and big diagrams

Together with zooming also the drawing of nodes was improved so that when there are too much nodes together their labels and even icons are hidden. They appear only when zoomed closer. Also the size of node changes accordingly.

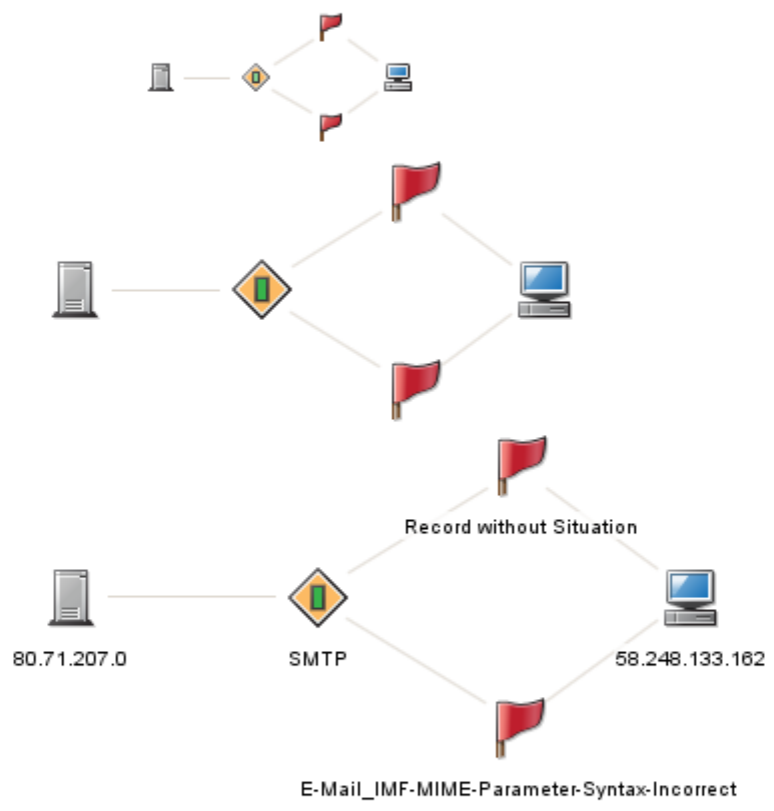


Image 8. Different sizes of nodes when zooming.

6.2 Progress Bar

Despite the fact that the event sets were limited, some operations can still take a lot of time. From the usability point of view, it is then very important to provide reliable indication about the progress. This must be taken into account in the early phase, because it is very hard to add it before hand. For example, the layout algorithm must be designed to include some kind of knowledge, how long it will proceed.

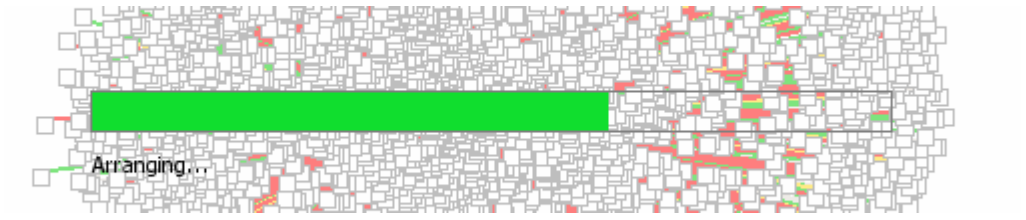


Image 9. Progress bar.

6.3 Accessibility

The visualization diagram itself is never enough for the user. There must be an easy to way to access details and to find more data. Therefore all the components in the diagram were made accessible. In other words, the user is able to select nodes and edges. When a component or a set of component is selected, additional details of them are shown in the visualization view.

Moreover, also mere hovering of components in diagram can provide additional details. This is essential when the overall structure is browsed where labels of nodes are not visible.

After several GUI prototypes a need for highlighting of paths was identified. In a complex visualization, it is not easy to see where edges are going and how nodes are linked together. Therefore a path highlight feature was implemented. When a node is hovered, also the links which are associated to it are highlighted.

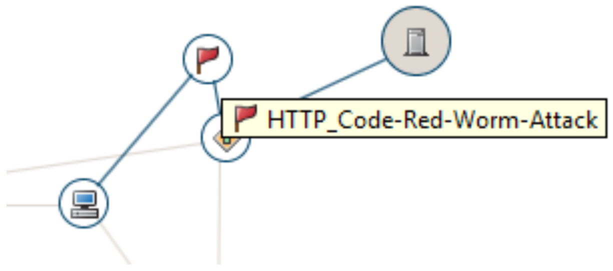


Image 10. Tooltip and connection highlighting for a node.

7 Conclusions

This report has covered some of the main design topics for implementing a working event visualization tool. One of the key finding has been that creating a user friendly tool requires a lot of attention to small details. Many of those small things can have a significant effect to the way data is gathered and processed.

The tool has been used for many different data sets. For an experienced administrator who knows the environment very well, the tool provides real added value by:

1) Visualizing usual patterns
2) Highlighting anomalies

This makes trouble shooting and incident investigation tasks faster and efficient. Moreover, it supports monitoring of logs which gives a different view than statistical reports.

8 Bibliography

Fruchterman, Thomas M. J.; Reingold, Edward M. (1991). *Graph Drawing by Force-Directed Placement.* Software – Practice & Experience (Wiley) 21 (11): 1129–1164.

Kamada, Tomihisa; Kawai, Satoru (1989). *An algorithm for drawing general undirected graphs*. Information Processing Letters (Elsevier) 31 (1): 7–15.