LARGE GRAPH VISUALIZATION OF MILLIONS OF CONNECTIONS IN THE CERN CONTROL SYSTEM NETWORK TRAFFIC: ANALYSIS AND DESIGN OF ROUTING AND FIREWALL RULES WITH A NEW APPROACH

Luigi Gallerani, CERN, Geneva, Switzerland

Abstract

The CERN Technical Network (TN) TN was intended to be a network for accelerator and infrastructure operations. However, today, more than 60 million IP packets are routed every hour between the General Purpose Network (GPN) and the TN, involving more than 6000 different hosts. In order to improve the security of the accelerator control system, it is fundamental to understand the network traffic between the two networks and to define new appropriate routing and firewall rules without impacting operations. The complexity and huge size of the infrastructure and the number of protocols and services involved, have discouraged for years any attempt to understand and control the network traffic between the GPN and the TN. In this paper, we show a new way to solve the problem graphically. Combining the network traffic analysis with the use of large graph visualization algorithms we produced usable 2D large color topology maps of the network identifying the inter-relations of the control system machines and services, in a detail and clarity, not seen before.

INTRODUCTION

Improving the security between the GPN and the TN, where accelerator systems and control systems run, is a priority for the infrastructure experts in the CERN Beams Department, Control Group, and the Computer Security team.

The separation between the GPN and the TN is defined by routing rules controlled by the Network Operations database (NetOps or LanDB). Only GPN hosts that are TN-TRUSTED are able to connect to the TN, and only TN hosts that are GPN-EXPOSED can communicate with the GPN. From the CERN NetOps records, routing tables are generated and loaded in to the routers.

We have today more than 1000 machines that are in the TRUSTED or EXPOSED lists with a justified operational reason. Around 6000 machines are involved in the communications between the two networks. The TRUSTED – EXPOSED list mechanism is clearly too weak compared to the modern network protection standards. The goal is to have in place well defined firewall and routing rules and have full control on the traffic between TN and GPN. To achieve that on a large control system in operation, it is fundamental to and understand all the relations and dependencies between the system involved.

A good starting point is the record of the real traffic on routers.

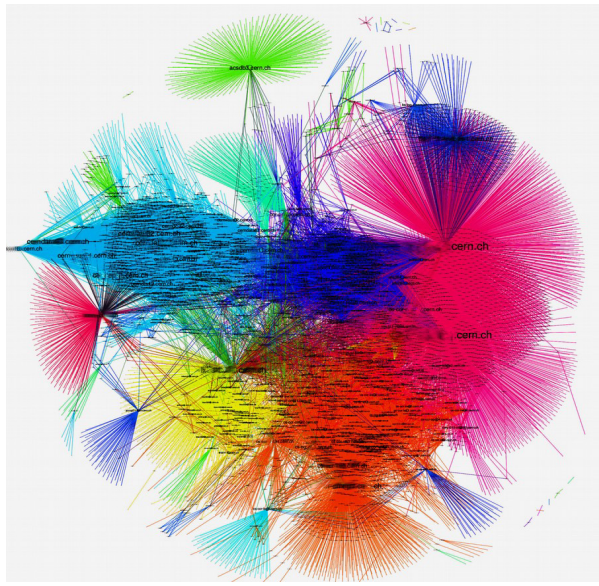

Figure 1: One hour of UDP-TCP traffic between CERN Technical Network and General Purpose Network plotted and clustered using Fruchterman Reingold algorithm.

RECORDING THE TRAFFIC

A large set of network restrictions rules can be easily defined by the knowledge of the system. However, this is not sufficient. With hundreds of different interconnected systems running on the TN and GPN, it is almost impossible to predict all the relations and dependencies between hosts and ports by the knowledge of the system functionality. The risk of compromising the connectivity of a running system because of a too strict or wrong firewall rule is what the Controls group wants to avoid most. Routers between the TN and GPN can record the traffic live producing as output text files with the following fields

Timestamp	Protocol	Packets count	Size bytes
Source host	Source port	Dest host	Dest port

DATA PROCESSING

Classical Approach by Statistics

In 1 hour, 60 million connections are recorded. 1.4 billion lines per day are the source of the analysis. Statistics, filtering, grouping and sorting are the obvious approaches for useful information extraction from a large dataset. For example it is easy to identify by queries the most connected hosts, the most used port between hosts of name N, the top 10 interconnected hosts and so on.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors

However, finding anomalies in the network connections, searching for hidden dependencies between multiple interconnected systems, getting the flow of traffic from histogram and results from statistics and queries are not easy or practical at all.

Graphs, Conversion in .dot Language

Graphs are the ideal mathematical tool to display host interconnections. Hosts are shown as a node N. Each connection is an edge E with source-dest ports as edge property. Packet size or the connection count is the weight of the edge. Timestamps can be used as the 4th dimension to display the evolution of the graph in time. Protocols such as TCP or UDP can be easily separated or plotted with different graphic like style or colored edges.

To give an example, two simple recorded connections between 3 host, two clients and one server, Figure 2 can be converted to Graphiz ".dot" language [1] with:

```
strict digraph G {
hostA -> servA [label="80 to 36642"
                bytes=85920 color="red" flows=50]
servA -> hostB [label="58139 to 46113" bytes=628
                color="blue" flows=6 ]
}
```

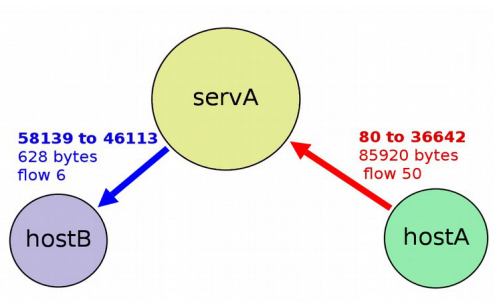

Figure 2: Direct graph automatically generated from the .dot language above using Graphiz and Gephi.

Graph Pre-Processor

Graphs directly generated from the recorded data have too many edges. Dot language interpreters automatically merge all the nodes entries with the same ID, but it is not immediate to do the same for the edges. We wrote a custom graph pre-processor script to perform the following operations and generate different types of sub-graphs in .dot format ready to be plotted.

1. Merge and count connections together hosts and port numbers (weighted edges)
2. Delete already defined connections (strict graph)
3. Ignore directions (undirected graph)
4. Separate in different graph for UDP and TCP

First Graph: Plotting 60M Nodes

To plot the obtained .dot graph "Gephi" [2] was used on "Debian 8" with "pekwm"[3] as X windows manager (Gephi is unstable on sophisticated windows managers like gnome/kde). Figure 3 shows the first graph obtained without edge labels and with random node positions. The complexity of the problem is evident, so additional steps are required to get better results.

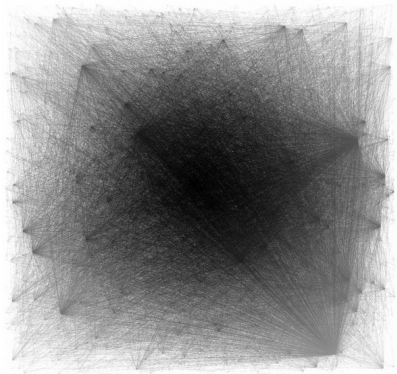

Figure 3: One hour of traffic: ~6K nodes with random position interconnected by 60M edges. Each node is an host, each edge is a TCP or UPD connction recorded.

GRAPH PROCESSING

Clustering and Neighbor Discovery

The first graphical computational step is called clustering [4] and Gephi already implements many different algorithms. Starting from a random node position graph like the one in Figure 3 a filter on nodes degree like K>3 is applied, then the Reingold Fruchterman [5] clustering process is started, and only at the end of the process, the filter is removed. A sequence of the clustering process is shown in Figure 4. The nodes with high degree K are kept in the center, and clusters of isolated hosts are moved to the edges.

Neighbours are discovered with the Chinese Whisper algorithm [6] so that nodes in the same cluster have the same colors and are easily identified. Finally, labels are applied. A full graph processing with this method is shown in Figure 1. In this article we show graphs with masked labels or replaced with generic ones.

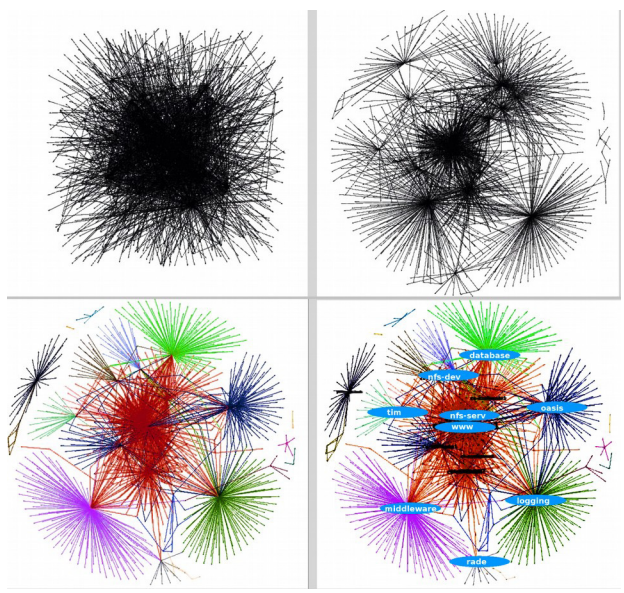

Figure 4: Clustering process sequence with Reingold Fruchterman algorithm followed by neighbor discovery colors by Chinese Whisper algorithm and label of nodes with high K degree.

Copyright © 2015 CC-BY-3.0 and by the respective authors

ISBN 978-3-95450-148-9

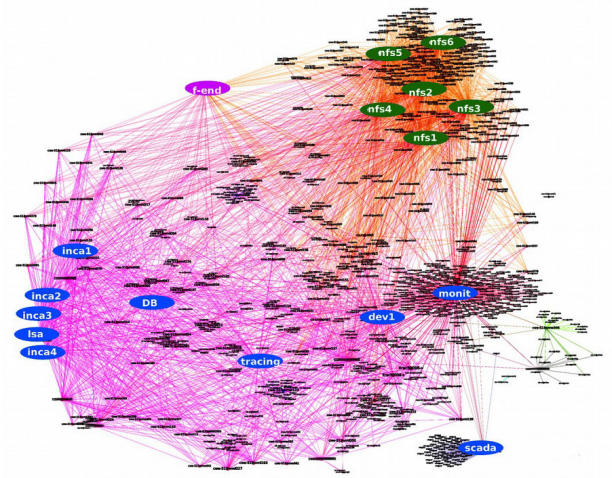

Figure 5: Cluster of around 400 nodes showing controls development machines dependencies in a strict direct graph over 24 hours in full resolution.

Sub-Graphs at Full Resolution

Once the main clusters of machines are identified, smaller and full detailed sub-graphs can be created. With the combined usage of the pre-processor and the filtering by queries inside Gephi, it is possible to generate full resolution sub-graphs like the one shown in Figure 5 where the TCP traffic generated in 24 hours by more than 400 development virtual machines is clearly plotted. To read all the information we print the high resolution graph in large format (ARCH E size). Using different pre-processor configurations, a full graph of port-to-port interconnections for same cluster is generated: Figure 6

Multi Graph Integration

Multiple graphs can also be combined together. A graph of host relations from NetOps has been combined with the recorded traffic graph. Hosts in the same network group are seen as a single large node. Connections of the similar

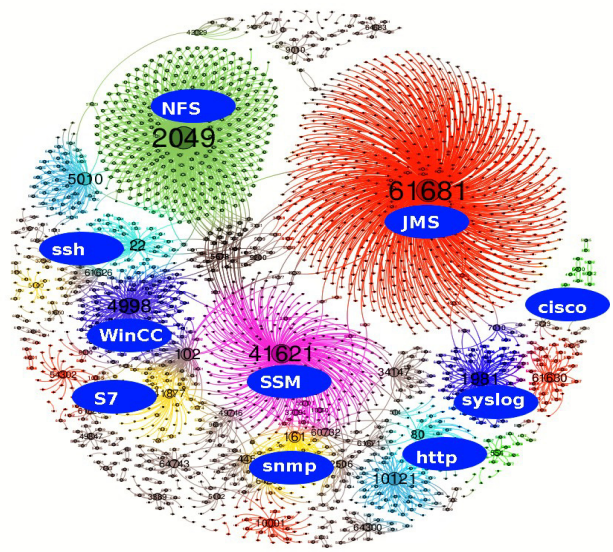

Figure 6: Port to port subgraph of the virtual machine cluster allows easily identifying the interconnections.

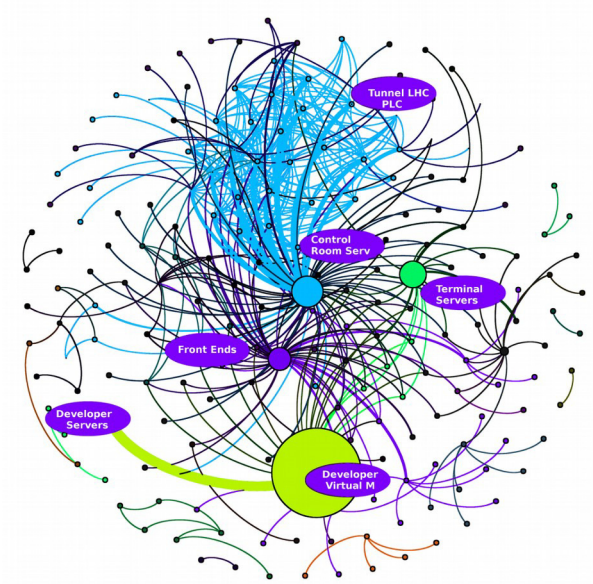

Figure 7: Combination of recorded data full resolution graphs with Network database info in a weighted edge graph where similar nodes are grouped together.

types are also grouped together in a larger edge line proportional to the enumeration, as visible in Figure 7

SECURITY FROM GRAPH ANALYSIS

Graph analysis returned highly valuable information that we converted in practical actions to improve the network security. For example, we identified and removed hidden dependencies for terminal servers, forgotten port scanners, wrong inclusions of development machines in operational PLC set. We moved the monitoring cluster out of the TN for more then 500 TRUSTED hosts. We have started to define firewall rules based on the analysis.

CONCLUSION

We used large graphs visualization techniques to plot million of recorded connections for machines involved in the controls system network. With the results we have been able to identify and fix many issues, have a view of all the dependencies and a new tool in place to define firewall and routing rules. This method can be easily adapted and widely used by the community involved in large control system network administration and protection.

REFERENCES

[1] Graphviz library www.graphviz.org
[2] Gephi Open Graph Viz platform gephi.github.io
[3] Pekwm Win manager www.pekwm.org
[4] V. Dongen, “Graph Clustering by Flow Simulation” PhD Thesis, University of Utrecht, The Netherlands.
[5] T. Fruchterman,;E. Reingold "Graph Drawing by Force-Directed Placement", Software – Practice & Experience (Wiley).
[6] C. Biemann, "Chinese Whispers an Efficient Graph Clustering Algorithm" University of Leipzig.

ISBN 978-3-95450-148-9

Copyright © 2015 CC-BY-3.0 and by the respective authors