2008

Malware, Viruses and Log Visualisation

Iain Swanson
Edith Cowan University

DOI: 10.4225/75/57b277c840cc3
Originally published in the Proceedings of the 6th Australian Digital Forensics Conference, Edith Cowan University, Perth Western Australia, December 3rd 2008.

Malware, Viruses and Log Visualisation

Iain Swanson

SECAU Security Research Centre
Edith Cowan University

Abstract

This paper will look at the current state of visualization in relation to mainly malware collector logs, network logs and the possibility of visualizing their payloads. We will show that this type of visualization of activity on the network can help us in the forensic investigation of the traffic, which may contain unwanted pieces of cod, and may identify any patterns within the traffic or payloads that might help us determine the nature of the traffic visually. We will further speculate on a framework that could be built which would be able to finger print any type of malware, based on the theory that the basic structure of Malware code does not change, it may mutate but the internal structure stays the same. By passing it through either a current log Visualisation algorithm or a purpose built piece of visual inspection software which would output a 3D visual representation of the malware to screen or be further processed by a multipoint mapping utility similar to a finger print mapper, which would determine the base structure of the malware and categorise it. If we could finger print zero day virus by recognising visually, we may then able to detect and create an antidote to it much quicker and more efficiently than is currently being done by most antivirus vendors

Keywords

Malware, Vizualisation, Network Security,

INTRODUCTION

The amount of data collected from a network, be it logs from the router, the proxy, the web server or from a malware collector such as mwcollect, ("Mwcollect," 2008) nepenthes,(Nepenthesdev, 2008) or honeytrap ("Honeytrap," 2008) the quantity of data and size of files can be daunting. If the data has come form a large corporation the log files could run into tens of gigabytes of data, this data has to be painstakingly searched through by someone looking of anomalies or in the case of honeypots looking where the traffic came from, identify the payload and then determine if it is malicious or not. According to (Frie & Rennhard, 2008) humans are very good at picking up and detecting patterns and analyzing images rather just text. Therefore the ability to see log files as a visual representation on the data contained within in it would greatly speed up the time required to analyze log files. The fact that humans can interpret complex visual images much faster than the text contained in the logs should bring visualization to the forefront of network forensics taking much of the tedious and painful trolling through data away as the examiner should be able to pinpoint anomalies and suspicious behaviors just by looking and the image that the data makes. Taking this another step forward the possibility of looking for and visualizing a virus or malware code in the same way would be quite possible, but what does it look like? Considering that it is only code it does not look like anything unless you are in The Matrix (Wachowski & Wachowski, 1999).

Malicious malware can be abstracted and rendered into another form as with Wechsler & Wrights description of their Digital Wilit software “the binary code of the virus is rendered as a sound wave; sound then becomes the input signal for a computer screen”(Wechsler & Wright, 2000). This solution is complex but gives the reader the idea of what can be done. Very little has change since 2000 in the representation of malware, but several applications have been developed to visualize log files which can produce very accurate representations of the details contained the log files and also incorporate the type of virus or malware which was included in the traffic. This paper it will look at some of the visualization programs and the different outputs that they produce and speculate on how they could be used and/or bolted together to produce a live on screen image of what is passing over the network to our computers before an antivirus product even knows they are there or at least has decided what it is. If a better way of identifying a piece of code was quicker than the current signature base methods, then zero-day virus could be caught and dealt with almost instantly. Although the proposal is of a signature nature it is creating more of a finger print which defines the malware by its internal structure and hopefully separates itself from signature bases detectors by it speed and accuracy of detection through the image it produces.

Assumptions

To visualise log files we need a large amount of computing power at our disposal for large files,and for a representation of malware we would need to off load some graphic processing to other machines, or better graphic cards if we want to see anything that resembles a pattern or an image in reasonable amount of time. Also there is the assumption that malware code has a particular internal structure, depending what type of malware it is.

VISUALIZATION

There are several types of visualization tools that can be used today to produce a visual representation of log files, although the appliance they have come from and their file format is a hindrance at this point in time, that being said many files can be analysed in this way. A framework by (Frie & Rennhard, 2008) is tiring to correct the file format problem amongst other things, by building HMAT (Histogram Matrix) which will be able to visualize any log format presented to it. The main problem at the moment is not that visualization tools do not work, they do work, but they are hard to use and consist of several steps using other numerous applications and consume a lot time and even more computing power.

Visualization comes into its own when we are talking about large files, we can analyse a 200mb file in a hex editor with relative ease but it still does not give us a good picture of the structure of the log its contents or a relationship between data in the logs, for example groups of IP addresses are not easily seen in a hex editor. There are different variants of tools which read a log file and produce different types of graphic representations, some tools will produce a 3D graphic of the logs which can also be broken down into sub sections like scatter plots, hemisphere spatial views, Axis Views and Total 3D views, at the moment 2D views are the most prevalent for looking at log files, determining what is happening on the network and where, just by looking at the 2D topology of the log representation. Several tools were looked at during this research to understand what the output was and how it could be adapted to view live networks or malware code.

Current Visualization tools

Here is a short list of current visualization applications:

- AfterGlow (Christian & Raffy, 2007)
- TreeMap (HCIL, 2008)
- InteVis (Van Riel, 2007)
- Cytoscape (UCSD, 2008)
- GraphViz (Low, 2004)
- TNV (Goodall, 2007)
- NvisionIP (NCSA, 2004)
- Rumint (G. Conti, 2007)
- Nazar (Roth, 2008)
- Skyrails (Widjaja, 2007)
- Sequoia (Bruls, Geerlings, & Van Ham, 2002)

There are many more applications that can visualize data but for this paper, some of above applications are the ones that were looked at. Some of the above applications work together to form a representation and others are stand alone applications that can generate a graphic with out any interaction with other software, for example to produce an image from a data stream we need to format it into a *.csv file, the following steps would have to be undertaken:

“\tcpdump -vttttnneli ath0 | \ ./tcpdump2csv.pl "sip dip dport" | head -2000 | \../graph/afterglow.pl -c color.properties -e 2 | neato -Tgif -o test.gif;”

This would generate a representation of the log generated by tcpdump, as can be seen here it is not the easiest or quickest way to get an image, also several different applications were involved, tcpdump, tcpdum2csv, afterglow and neato. However, it would still be remarkably quicker than searching through the log files manually to detect any anomalies. Some of the tools above, although not specifically designed to visualize network logs or malware identification they can be manipulated to do so.

Visualization at work

The following section will demonstrate some of the techniques and representations that are possible with the applications and discus what can be learnt form the out put of each application. (Blasco, 2005) takes a Nepenthes log file, turns the log into a CSV file and generates a visualization of the malware found in the log file. Finding the hash value of the malware he manages to produce this image:

```
# cat datos.csv | perl afterglow/src/perl/graph/afterglow.pl -c color.properties -e 6 -p 1 > img.dot
# cat img.dot | neato -Tgif -o test.gif
```

Image generated:

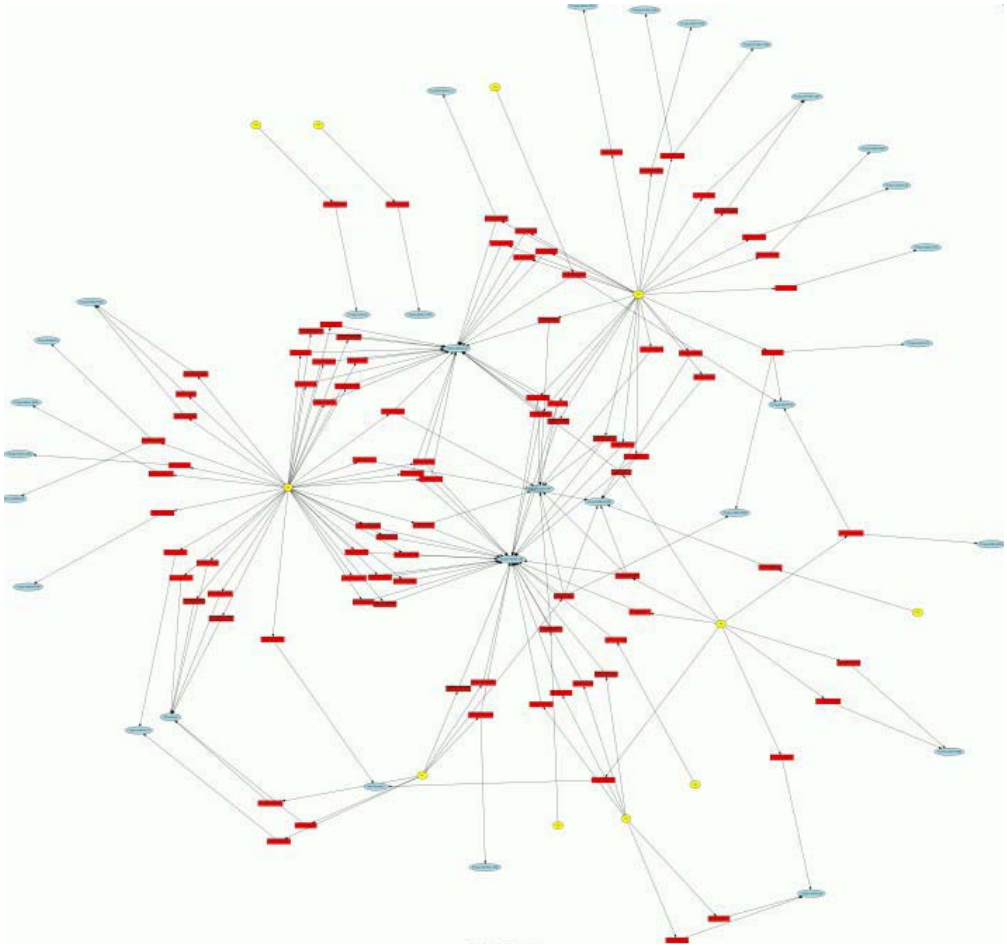
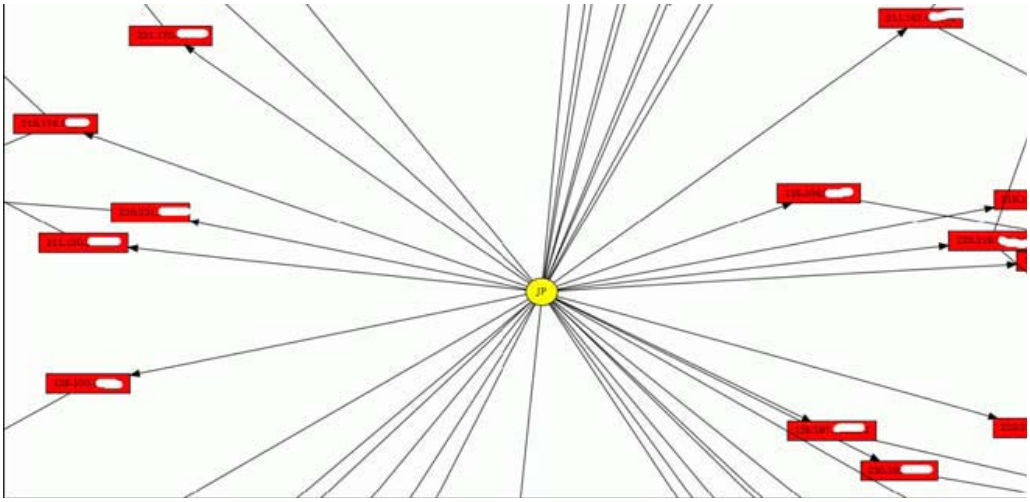


Figure 1 total image of malware spread (Blasco, 2005)



Figurev2 closer view of image(Blasco, 2005)

We can now see ip addresses appearing, giving more of an idea of where things are moving ‘from’ and where they are going ‘to’ generally countries but can be narrowed done to ISP ip ranges therefore identifying the culprit is possible.

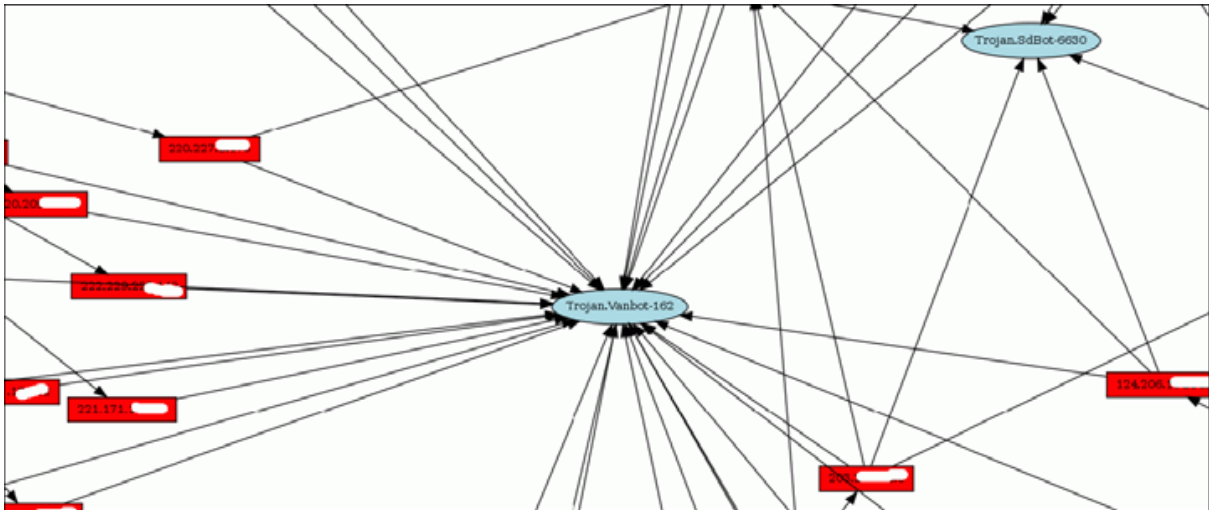
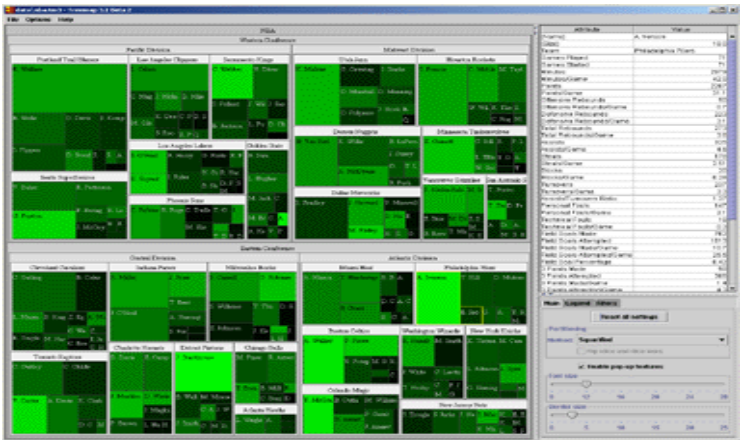


Figure 3 closer inspection (Blasco, 2005)

TreeMap (HCIL, 2008)

“Treemap is a space-constrained visualization of hierarchical structures”



SequoiaView(Bruls et al., 2002)
Similar to TreeMap but “Squarified treemaps”. “The screen is subdivided such that rectangles approach squares as closely as possible”

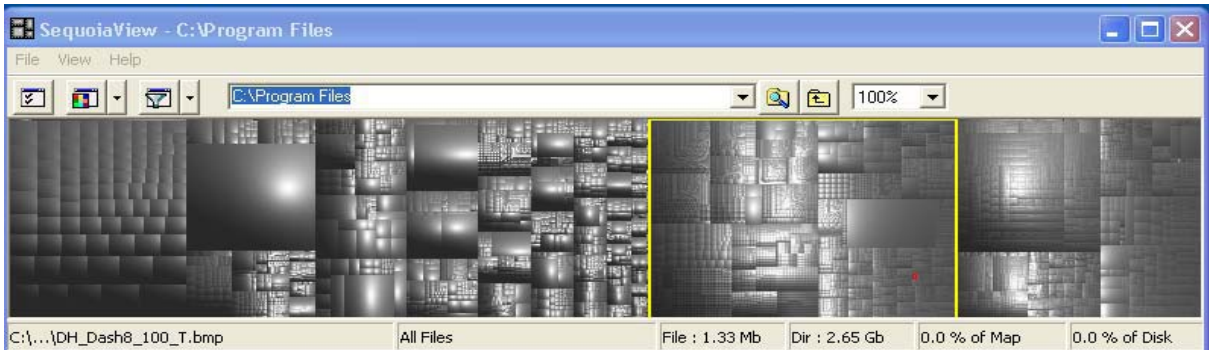


Figure 4 SequoiaView (Bruls et al., 2002)

InetVis(Van Riel, 2007)
InetVis is a 3-D scatter-plot visualization for network traffic. Taken for the “The Spinning Cube of Potential Doom". “The vast majority of coloured dots can be considered to be malicious traffic searching for vulnerable systems” (Paxson, 2003). Other network visualizations employ lines as a metaphor for connection, the 3-D scatter-plot of points proves to scale well with larger volumes of data.(Van Riel, 2007).

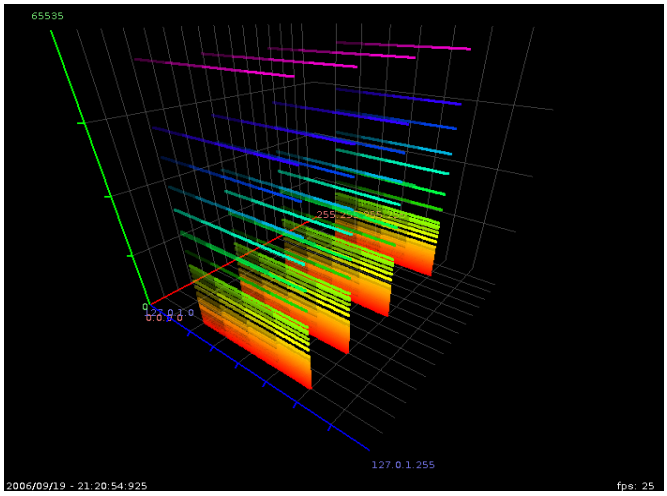


Figure 5 InetVis(Van Riel, 2007)

“The visualization below is from Tenable Network Security's, Security Center, which includes a 3D visualization tool that can derive network topology information from distributed Nessus vulnerability scanners. Each node in the center helix of the above graph is detected router”(Tenable Network Security, 2008)

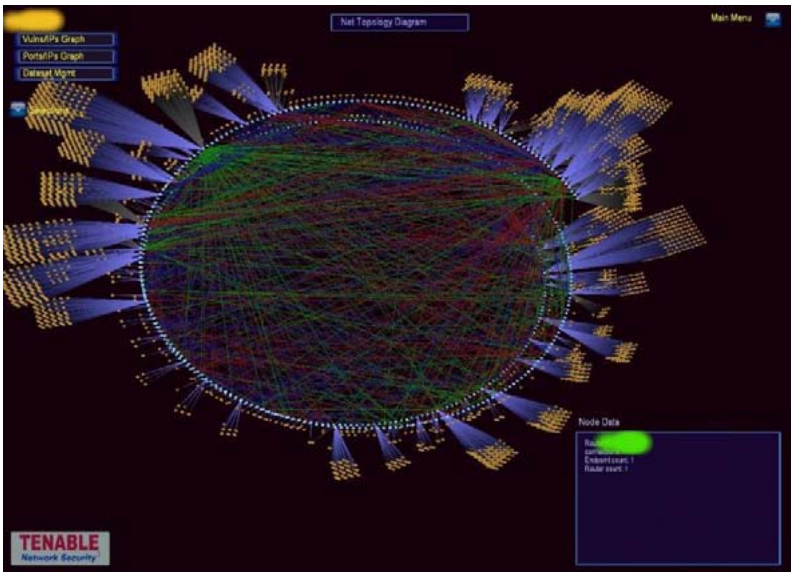


Figure 6 (Tenable Network Security, 2008)

Skyrails (Widjaja, 2007)
"Skyrails is a Social Network and Graph Visualization System with a built-in programming language" This can be programmed to input any log files and configured to render in different ways.

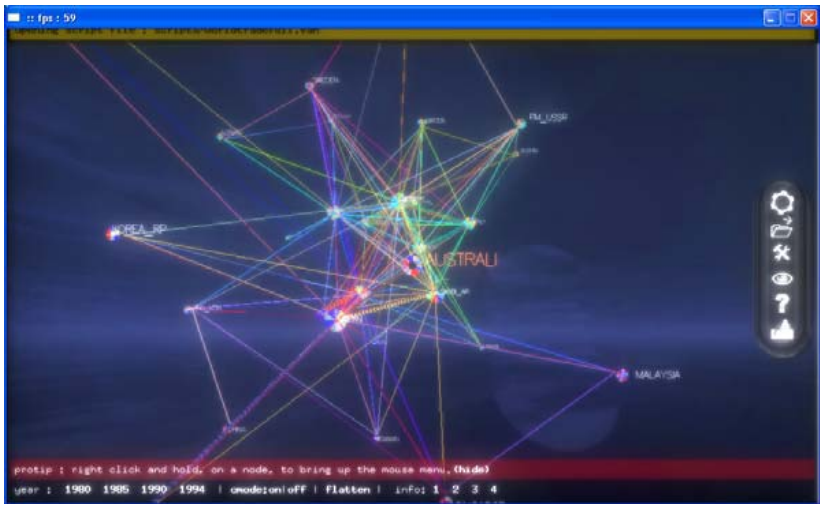


Figure 7 Skyrails (Widjaja, 2007)

As can be seen for the images and software previously mentioned, the techniques, inputs an rendered outputs are vastly different from each other but all have the same aim. That aim is to see the coded world as a graphic representation. All the programs have a level of complexity that needs to be rectified if the tools are to become more useful. Raffael Marty and Jan P Monsch have brought to the arena a tool that encompass many of the current tool sets in one easy to use (relatively speaking) LiveCD. The DAVIX (Data Analysis and Visualization Linux), tools "provide an integrated out-of-the-box environment for data and visualization analysis"(Marty & Monsch, 2008). As noted by Dr Greg Conti "For InfoSec practitioners just starting out in visualization, the best place to start is by experimenting with existing tools. Unfortunately, finding and correctly installing these tools can be a tricky process. That is why I'm excited by the DAVIX project. It integrates a wide range of tools into one easy to use distribution."(Conti, 2007). The DAVIX collection contains about 25 different and complementing visualizing tools which are outside the scope of this paper to discuss all the tools in any great detail but some are mentioned in this paper for example, 'afterglow' suffice to say that having all visualizations tools in one environment makes the task much easier.

Conceptual Proposal

Background in Antivirus detection:

There are several ways a detector can detect malware or viruses these are scanning, integrity checking, interception, and heuristic detection. The pros and cons are listed below, taken form (Roberts, 2001)

- Scanning: “Scans all files and uses a signature to detect the malware, There are two major disadvantages to scanning-based techniques. Firstly if the software is using a signature string to detect the virus, all a virus writer would have to do is modify the signature string to develop a new virus. This is seen in polymorphic viruses. The other, and far greater disadvantage is the limitation that a scanner can only scan for something it has the signature of.”

- Integrity Checking: “Uses Check summing, problem is with integrity checking is that not enough companies offer comprehensive integrity checking software.”

- Heuristic Detection: “This is a generic method of virus detection. Anti-virus software makers develop a set of rules to distinguish viruses from non-viruses. Should a program or code segment follow these rules, then it is marked a virus and dealt with accordingly.” The problems with this are “the technology today is not sufficient, and virus writers can easily write viruses that do not obey the rules, making the current set of virus detection rules obsolete.”

- Interception: “Interception software detects virus-like behaviour? Interceptors are not very good at detecting anything else. Interceptors have all the drawbacks of heuristic systems – difficulty differentiating virus from non-virus, and easy to program around.”

Based on a project by Alex Dragulescut of visualization of worms, viruses, trojans and spyware code. This paper proposes a theoretical solution to both network logs and malware verification in a timely manner. Although Dragulescuts images are art, “each piece of disassembled code, API calls, memory addresses and subroutines are tracked and analyzed. Their frequency, density and grouping are mapped to the inputs of an algorithm that grows a virtual 3D entity. Therefore the patterns and rhythms found in the data drive the configuration of the artificial organism.”(Dragulescut, 2008). Below are the theoretical structures of malware and viruses.

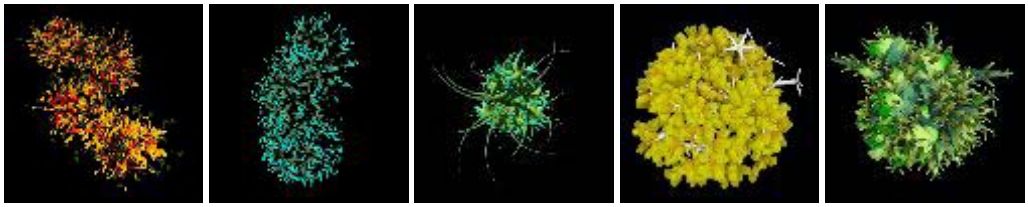


Figure 8 (Dragulescut, 2008)

from left to right: PWSLineage, Stormy, MyDoom, IRCbot, Virutmytob

Skyrails is a highly developed and visual tool that can render in 3D a DeepWorld© type of visualisation, showing what traffic is on the network, where it is going and pinpointing suspected anomalies in the traffic. The framework can be programmed to automatically search the traffic and discover suspect packets, extract the data and pass it to Dragulescuts organic algorithm. This will create the image, show it on screen or pass it to a fingerprint pattern classification application. This applications will map the pre determined minutiae points and derive the results then alert the administrator, all data collected would be stored in a large database for future analysis.

This could also be on screen via large LCD displayed in the NOC (network operations centre) monitored by staff.

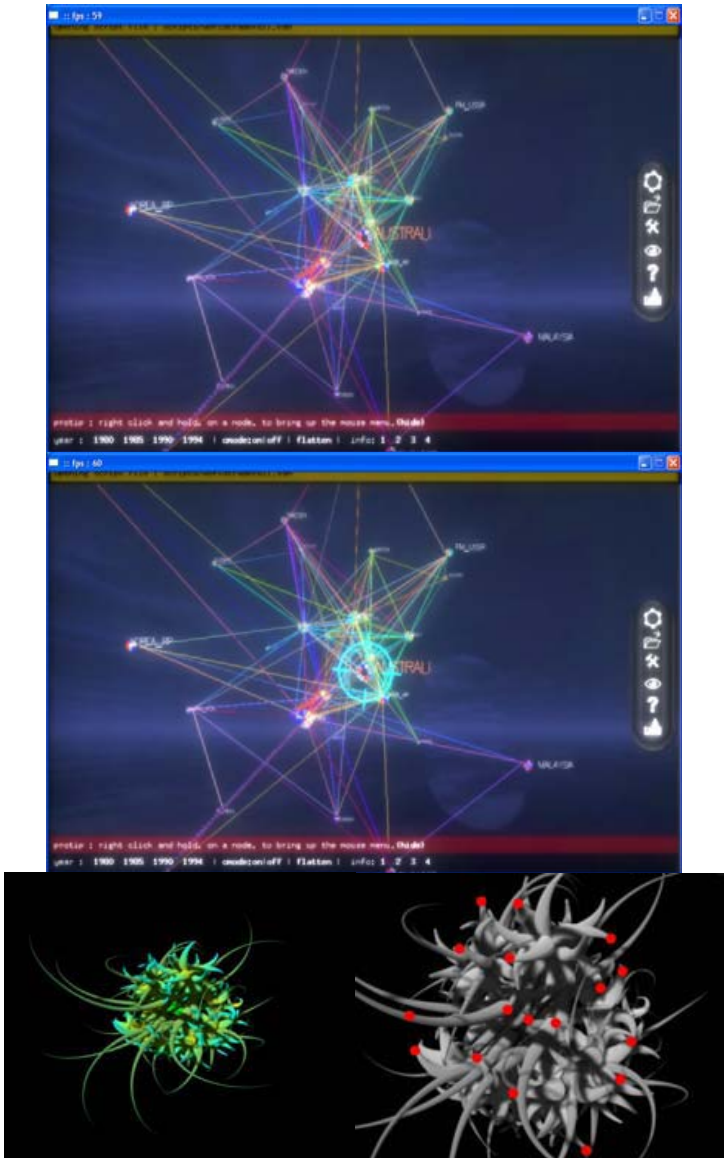


Figure 9 example of identification

The Scenario above see's Skyrails scanner watching the traffic on the network and either automatically or manually the anomaly is located, the code is extracted and passed to Dragulescuts organic algorithm which renders the code as an image. Which can then be displayed on a screen, staff can immediately identify the image on the screen as a virus or malware by it unique structure viewed as a recognisable image, steps can then be taken to deal with the anomaly, or it can be passed for verification to the pattern classification application for analysis and quarantined. This system could be rebuilt into one application and installed onto computer as part or instead of the traditional antivirus protection currently out in the market place.

CONCLUSION

The use of visualization can clearly be seen as a better solution to the endless task searching through logs by visualising the activity on the network. Is it better to use 3D or 2D? The belief is that 2D images using graph base representations although very useful do not scale well and have mapping and layout problems. It can be seen from the research in this paper that all applications can visualize data, some better than others in speed and accuracy but most can be reconfigured to perform different tasks in the identification of malware, where it came from and what patterns it forms if any.

This paper was intended to be an introduction and theriacal paper on visualization and has not drilled down into specific detail of application API's, storage backend or looked at tiering models, such as Application tiering, Model tiering and Scan tiering discuses in the paper "A Framework for Unified Network Security Management:

Identifying and Tracking Security Threats on Converged Networks" (Dawkins, Clark, Manes, & Papa, 2005). More research needs to be done into how it would be possible to visualise malware more accurately and not rely on signatures, this paper believes this to be possible as all signs point to malware code being unique much like DNA Genome mapping. InSeon Yoo has done research on this and has come up with, Self-Organizing Map (SOM) "this is an unsupervised neural network method which has properties of both vector quantization and vector projection algorithms" (Yoo, 2004). This can detect viruses inside windows executable files without the aid of signatures by visualizing the virus code inside the executable. The future of visualization of either log file analysis or malware payloads within a network lie with 3D graphic images being displayed as interactive DeepWorld© visualization which can be drilled deep to identify anything that is happening, not just within suspect code or packet but the whole network for congestion or forensic analysis.

As mentioned in Greg Conti's book, Security Data Visualization: Graphical Techniques for Network Analysis, on the back cover, "a picture is worth a thousand packets"(Conti, 2007). In this researchers opinion it would probably be more like "a picture is worth million packets".

ACKNOWLEDGEMENT

This paper is loosely based on an idea from a lecture give at University of Glamorgan, UK and presented at the 7th Annual IEEE Information Assurance Workshop about "Visualization Framework for Intrusion Detection using three tiers, Visualisation, Middleware, and Database". (Read & Blyth, 2006)

REFERENCES

Blasco, J. (2005). An approach to malware collection log visualization. *Journal*. Retrieved from http://www.secviz.org/

Bruls, M., Geerlings, J., & Van Ham, F. (2002). *SequoiaView*, from http://w3.win.tue.nl/nl/onderzoek/onderzoek_informatica/visualization/sequoiaview//

Christian, & Raffy. (2007). *AfterGlow* 1.5.9. from http://afterglow.sourceforge.net/

Conti (2007). Security Data Visualization, Network Security Available from http://nostarch.com/securityvisualization.htm

Conti, G. (2007). *Rumint* 2.14. from http://www.rumint.org/

Dawkins, J., Clark, K., Manes, G., & Papa, M. (2005). A Framework for Unified Network Security Management: Identifying and Tracking Security Threats on Converged Networks. *Journal of Network and Systems Management,*.

Dragulescut, A. (2008). *Malwarez, a series of visualization of worms*, from http://www.sq.ro/index.php

Frie, A., & Rennhard, M. (2008). *Histogram Matrix: Log File Visualization for Anomaly Detection*. Paper presented at the International Conference on Availability, Reliability and Security. from http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4529398

Goodall, J. (2007). *TNV (The Network Visualizer or Time-based Network Visualizer)* 0.3.8. from http://tnv.sourceforge.net/

HCIL. (2008). *TreeMap* 4.1.1. from http://www.cs.umd.edu/hcil/treemap/

Honeytrap. (2008). *Honeytrap*, from http://honeytrap.sf.net/

Low, G. (2004). *GraphViz* 0.99. from http://www.graphviz.org/About.php

Marty, R., & Monsch, J. (2008). *DAVIX* 1. from http://www.secviz.org/content/the-davix-live-cd

Mwcollect. (2008). *Mwcollect*, from http://www.mwcollect.org/

NCSA. (2004). *NVisionIP*, from http://security.ncsa.uiuc.edu/distribution/NVisionIPDownLoad.html#NVisIP

Nepenthesdev. (2008). *Nepenthes* 0.2.2. from http://nepenthes.mwcollect.org/

Paxson, V. (2003). *The Spinning Cube of Potential Doom*. Paper presented at the SC03. from http://www.nersc.gov/nusers/security/TheSpinningCube.php

Read, H., & Blyth, A. (2006). An Integrated Visualisation Framework for Intrusion Detection. *Journal*. Retrieved from http://www.itoc.usma.edu/Workshop/2006/Program/Presentations/IAW2006-13-2.pdf

Roberts, E. (2001). Computers, Ethics, And Social Responsibility. *Journal*. Retrieved from http://cse.stanford.edu/class/cs201/Projects/viruses/anti-virus.html

Roth, F. (2008). *Nazar* 1. from http://nazar-interface.blogspot.com/

Tenable Network Security (2008). Security Center Security Center Available from http://www.tenablesecurity.com/solutions/

UCSD. (2008). *Cytoscape* 2.6.0. from http://www.cytoscape.org

Van Riel, J. (2007). *InteVis* 0.9.5. from http://www.cs.ru.ac.za/research/g02v2468/inetvis/0.9.3/doc/inetvisdoc.html#1.

Wachowski, A., & Wachowski, L. (Writer) (1999). *The Matrix*.

Wechsler, L., & Wright, A. (2000). Digital Wilt: Art from a Computer Virus. *Journal*. Retrieved from http://www.teigig.net/papers/visFinal.pdf

Widjaja, Y. (2007). *SkyRails* 0.1. from http://cgi.cse.unsw.edu.au/~wyos/skyrails/index.php

Yoo, I. (2004). Visualizing Windows Executable Viruses Using Self-Organizing Maps. *Journal*. Retrieved from http://vx.netlux.org/lib/aiy00.html

COPYRIGHT