HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Information and Natural Sciences
Department of Computer Science and Engineering

Zhihua Jin

Visualization of Network Traffic to Detect Malicious Network Activity

Master's Thesis

Espoo, June 30, 2008

Supervisors: Professor Antti Ylä-Jääski, Helsinki University of Technology
Professor Yuming Jiang, Norwegian University of Science and Technology

Instructor: Morten Knutsen, M.Sc, UNINETT AS
Arne Øslebø, M.Sc., UNINETT AS

HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Information and Natural Sciences
Degree Programme of Security and Mobile Computing

ABSTRACT OF MASTER'S THESIS

Author Zhihua Jin

Date June 30 2008

Pages 102

Title of thesis

Visualization of Network Traffic to Detect Malicious Network Activity

Professorship
Data Communications Software

Professorship Code
T-110

Supervisors
Professor Antti Ylä-Jääski
Professor Yuming Jiang

Instructor
Morten Knutsen, Arne Øslebø

Today, enormous logging data monitoring the traffics of the Internet is generated everyday. However, the network administrators still have very limited insight into the logging data, mainly due to the lack of efficient analyzing approaches. Most of the existing network monitoring or analysis tools either mainly focus on the throughput of the network in order to assist network structure planning and optimization, which is too high level for security analysis, or dig to too low level into every packet, which is too inefficient in practice.

Unfortunately, not all network traffics are legitimate. As a matter of fact, a lot of malicious traffics flow through the Internet all the time. Such malicious traffics can lead to various cyber-crimes, and exhaust considerable network bandwidth. The expression that what you do not see can hurt you perfectly suits the situation here.

In order to help the network administrators to discover malicious activities in their network traffics, this thesis attempt to explore suitable visualization techniques to distinguish malicious traffics from massive background traffics by using visual patterns, to which the human visual perception system is sensitive and can thus processes efficiently.

To achieve such goal, we first extract the visual patterns of malicious activities from known malicious traffics. Then, we look for the same visual patterns in the normal traffics. When the same visual pattern is found, we identify the relevant malicious activities.

The tool used in our experimentation is designed and implemented according to the experiences learned from previous related works, with special regards to human visual perception theory. The result of our experimentation shows that some malicious activities which can not be easily identified in traditional analyzing approaches before, can be identified by our visualization system under certain conditions.

Keywords security visualization, netflow, malicious activity detection

Acknowledgments

This thesis work was carried out at the Department of Telematics at The Norwegian University of Science and Technology (NTNU).

First I would like to thank my supervisor Prof. Yuming Jiang at NTNU and co-supervisor Prof. Antti Ylä-Jääski at the Helsinki University of Technology (TKK) for their guidance at a high level of my work. As the main supervisor, Prof. Yuming Jiang introduced me to UNINETT, which is the actual place I carried out my work. During this work, he gave me many ideas and valuable advices in several close discussions with me. In the end, he helped me to prepare an abstract paper of this work for a conference.

I would also like to thank my instructors Morten Knutsen and Arne Øslebø at UNINETT. Morten gave me a lot of good advice in the practical work such as the implementation of our own tool and the experimentation of detecting malicious activities in their NetFlow logs. Arne help me to get acquainted with the NetFlow monitoring facility of UNINETT, which is very important in this work as well. In addition I appreciate UNINETT for providing me their NetFlow monitoring logs and the computing facilities.

Lastly I would like to thank Dr. Guoqiang Hu for valuable advices on revision of this report, and my friend Hanbo Zhao, Kaiyu Dai and Ana Hristova for many practical helps in my work and the good time together. Without you this work can not be completed.

Espoo, July 2008

Zhihua Jin

Abbreviations and Acronyms

ACK	Acknowledgment
AI	Artificial Intelligence
AS	Autonomous System
BPP	Byte per Packet
BYT	Byte
CERT	Computer Emergency Readiness Team
CLI	Command Line Interface
CPU	Central Processing Unit
CSV	Comma Separated Value
CTA	Cognitive Task Analysis
C&C	Command and Control
DA	Destination Address
DAS	Destination Address String
DoS	Denial of Service
DDoS	Distributed Denial of Service
DP	Destination Port
DST	Destination
FFT	Fast Fourier Transform
FIN	Finish
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
IPFIX	Internet Protocol Flow Information eXport
IRC	Internet Relay Chat
ISP	Internet Service Provider
JVM	Java Virtual Machine

JNI	Java Native Interface
MDMV	MultiDimensional MultiVariate
MDS	MultiDimensional Scaling
MRTG	Multi Router Traffic Grapher
OS	Operating System
OSI	Open Systems Interaction
PCA	Principle Component Analysis
PCP	Parallel Coordinate Plot
PKT	Packet
RAM	Random Access Memory
RRD	Round-Robin Database
SA	Source Address
SAS	Source Address String
SNMP	Simple Network Management Protocol
SP	Source Port
SRC	Source
SYN	Synchronize
TCP	Transmission Control Protocol
TD	Time Duration
TE	Time End
TES	Time End String
TOS	Type of Service
TS	Time Start
TSS	Time Start String
UDP	User Datagram Protocol
VCR	VideoCassette Recorder
VizSEC	VisualiZation of SECurity

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Without doubt, the Internet nowadays is becoming bigger in size and faster in speed. With the support of technology development, the Internet extends wider and penetrates deeper in our life. As a result, the amount of its traffics grows explosively, and the variety of the network services increases rapidly as well. Such enormous amount and complex composition of traffic pose a big challenge for the network management, especially for the network security management.

In order to manage a network, the traffics going through it have to be well understood. In other words, the network traffics have to be measured and analyzed. For the traffic measurement, there are two ways in general. One is to capture only the network properties of the traffics without the payload, normally performed by the network routers and the captured data are stored as network flow records. The Cisco NetFlow[1] is a typical technology of this type. The other one is to capture all information of the traffics including the payload, normally performed directly over the network interface driver and the captured data are stored as network packet records. The TCPDUMP/LIBPCAP[2] is a typical technology of this type.

Apparently, the packet based methods contain more information than the flow based methods, which should be preferred for more thorough analysis. However, it is usually not possible to apply the former type of methods because of legal issues. Also, more information means higher maintainenance and analysis costs. As a matter of fact, even for the flow based methods, the utilization of the captured data is far below its full potential and most of the time the data remain untouched after a few basic analysis at the very beginning. In addition, because the Internet traffics can be encrypted as the IPv6 emerges, the payloads contained in the data captured by the packet based methods are going to lose their initial advantages. Therefore in practice, the flow based methods usually come as the first choice, so doe they in this work. More specifically, we use the Cisco NetFlow technology in this work.

Even though only partial information of the network traffics is obtained by applying the flow based method, the resulting data are still huge. Thirty giga-bytes captured flow data per day is considered very common in the backbone networks of medium or large Internet service providers (ISPs). On one hand, such tremendous data may contain a lot of useful information that could help better understand the traffics. On the other hand, such enormous data certainly require very efficient analyzing methods and lots of analyzing resources.

Using the existing analyzing tools, which are mostly text based and provide very basic analyzing functions, the network administrators can perform quite efficient analysis but come up with few useful results. The limited functionality and the poor user interface of the tools lead to the unsatisfactory utilization of the captured data.

However, as malicious activities never stop and tend to increase along the growth of the Internet, they are always a major threat to the Internet security. As warned by Russ McRee in [3] that “What you don't see can hurt you”, network administrators had better try other ways to gain more

insights into their networks in order to make correct decisions timely. Visualization seems to be a suitable way because comparing to the text based methods: 1) it can usually show much more information in the same viewing space[4]; 2) it is possible to be processed very efficiently by human beings by making use of human perception capabilities[5]; and 3) many times in history visualization has shown unique capabilities in assisting from exploratory analysis to decision making[6], e.g., its successful location of the contaminated source in the cholera epidemics in London in 1854.

In this work, we try to apply visualization methods to assist the network administrators in the analysis of the netflow data collected from real core Internet. The outcome is a prototype tool named UniVis that visualizes the netflow data as requested during the interactions with user. UniVis should enable fast identification of the network anomalies in the general overview, convenient investigation into interesting spot for more details, and ultimately help the administrators gain comprehensive understanding of their networks and hence hopefully make more informative decisions and more timely responses during the network security management.

In order to achieve such ultimate goal, more approaches are needed to enhance the visualization tools. As mentioned earlier, visualization mainly improves the efficiency of analysis by increasing the amount of information being taken by human users in each single process. With the help of proper human-computer interaction techniques, the interpretation of the visualization could be enhanced by establishing domain-specific semantic relationships during the exploration. However, how to interpret the visualization, or rather, how to correlate the visual interests with actual network anomalies, remains the job of we human users. In this work, experimentations of using our visualization tool to explore network anomalies are carried out in order to acquire such experiences.

1.2 Scope

Though information visualization involves computer graphic theory and technology, which is about specific computer drawing techniques, it is not the main issue in this work and such content is not concerned. As a matter of fact, rather than focus on the drawing details, we focus more on the visualization methods at a higher level, which is about the form in which they visually present the data.

Besides direct visualization of the netflow data, there are also many algorithm based methods to enhance and highlight the visual patterns which raise most concerns. For example, multidimensional reduction algorithms could be used to classify the data and isolate the outliers. In addition, spectrum analysis could be used to identify spectrum patterns of the data, whereby correlation analysis could be used to establish relationships between similar temporal or spacial patterns. Although these algorithm based methods may throw lights on the future exploration, they are not the main direction in this work.

As stated in Section 1.1, the netflow data is chosen as the main input of our visualization. In other words, packet analysis is out of the scope of this work, though the results of packet analysis are used to identify some malicious activities at the first place and to confirm the anomalies discovered by the visualization methods in the end. In addition, as each set of input data is generated from one router, we only analyze the characteristics of the flows observed by this

single router, whereas the analysis concerning the relationships of traffic characteristics among multiple routers is not considered.

1.3 Criteria

Since this thesis work results in the UniVis, a prototype network traffic visualization tool built for assisting the visual exploration of network anomalies, some criteria should be introduced as a guideline both for the initial designing and the final evaluation of such solution.

Considering the solution a analysis tool, the functionality naturally becomes an important criterion. For example, how many visualization methods are supported, how many human-computer interaction techniques are featured, and so on. As a tool aiming at assisting the discovery of network anomalies, the variety of anomalies it can clearly represent, the efficiency of the exploration process it can support, and the accuracy of the identification of anomaly it can help to achieve are all essential criteria.

As a information visualization system, two major types of criteria must be taken into account. On one hand, how well does it follow the human cognition and perception theory in order to take most advantage of the human visual perception system. In other words, the human perception capabilities need to be enabled and optimized to absorb the information visually represented by the system. We name such ability perception efficiency in this work. On the other hand, because the visualization system and sometimes even the human perception system could become the targets of intentional attacks, the ability of resisting or evading from such attacks is also an important criterion, which is referred as robustness in this work.

In addition to the above crucial criteria, some common criteria in implementation and utilization practice are also applicable, such as scalability, extensibility, and usability. Because of the huge amount of input data, fast processing speed become very critical and thence scalability turned out to be the most concern of the network administrators.

1.4 Methodology

As the thesis title speaks for itself, this thesis work can be divided in two parts: visualization of the network traffic, and using the visualization to detect malicious network activities. For each part we have a methodology.

For the visualization of network traffic, we first make qualitative comparisons between several selected visualization methods, in order to find the most suitable ones according to the structure of the netflow data and the relevant criteria described in Section 1.3. Then, we make an exhaustive survey of existing visualization tools or toolkits, and the existing network security visualization solutions, and evaluate them with relevant criteria in Section 1.3. Based on the comparison and survey results in the previous two steps, we then start to design and implement our security visualization prototype – UniVis. Of course our prototype is then evaluated under the same criteria as those being used to evaluate other solutions.

As for the detection of malicious network activity, our methodology consists of two major steps. Firstly, we explore suspicious patterns in the visualizations of the unsampled netflow data, which

is captured from small enterprise networks that are known to contain malicious traffics. Such patterns should be confirmed to be caused by certain malicious activities if possible. Secondly, we try to find more instances that are similar to the confirmed patterns, from both the visualizations of the original data and the visualizations of the sampled netflow data which is captured from the backbone network during the same time window. If the same pattern caused by certain malicious activity can be found in the visualizations of the original data, it proves that those visualizations are effective in identifying such malicious activity. If the same pattern caused by certain malicious activity can be found in the visualizations of the sampled backbone network data, those visualization methods show great potential to be used to detect such malicious activity in practice, since most netflow data are sampled in practice.

1.5 Related Work

Research on applying information visualization techniques to computer security analysis has lasted for years. At least the most active annual event in this field – the VizSEC workshop[7] has went through 5 years since year 2004. During these years, various security visualization solutions are proposed. Although those solutions concentrate on various purposes and use different kinds of data, in this work we focus only on those visualizing the network traffics, especially those visualizing the netflow data. In Chapter 2 we will discuss more about them.

Unfortunately, there are not many solutions fall in our focus. Worse more, none of the existing implementation of the solutions for the netflow data takes the Cisco NetFlow data format, or fulfill our criteria well enough. Thus, we decided to build our own solution – UniVis instead of adopting an off-the-shelf one.

1.6 Report Outline

This report is divided into nine chapters and an appendix. The results of both the comparison of candidate visualization methods and the survey of existing solutions are given in Chapter 2 Technological background of the main components in our solution is introduced in Chapter 3, followed by Chapter 4 with a brief introduction of UNINETT, which provides the main using scenario for our solution. Chapter 5 explains the system design of our solution UniVis, while Chapter 6 presents the implementation of the prototype. In Chapter 7, the experimentation of utilizing the prototype to detect network anomalies is described. Then Chapter 8 have more discussion of the practical meaning of the experimentation results. Finally a conclusion is given in Chapter 9, with some directions of future work.

2 Related Work

Security visualization became a hot topic in recent years, when many researchers have dedicated themselves in the attempt to apply information visualization to the field of the computer security, in order to get more insights into the explosively increasing security data efficiently. Though the key words in such effort are information visualization, the key technologies and theories involved are way beyond the scope of the information visualization itself. A quick list of several main relevant disciplinary fields required by the successful achievement of such goal includes at least human perception and cognition theory, human-computer interface, data mining, computer system and network security, and computer graphics.

As one of the most active platform for the research community of security visualization, the annual VizSEC workshops have witnessed a lot of discoveries and novel solutions since it was held for the first time in 2004. In the latest happening in 2008, it covers an exhaustive problem set of the state-of-the-art in cyber security, such as visualization of Internet routing, visualization of packet traces and network flows, visualization of intrusion detection alerts, visualization of application processes, etc. In addition, it also starts to give more concern on issues of the VizSEC system itself, including deployment and field testing of VizSEC systems, evaluation and user testing of VizSEC systems, user and design requirements for VizSEC systems, and development of VizSEC systems. In the following of this chapter, we introduce and evaluate several representative solutions for the visualization of network traffics, and discuss some issues about the VizSEC system itself concerning our criteria.

2.1 Survey of Existing VizSEC Systems

The survey result of several VizSEC systems are presented in this section. In the end, Table 2.1 gives very brief comparison among these systems.

2.1.1 NfSen

In fact, NfSen[8] is not quite a VizSEC system but a network bandwidth usage analyzer with limited graphical supports. It is selected as the first in our survey because it is a typical traditional traffic analyzing tool, which derives from tools mainly use SNMP for data collection, but uses netflow data also.

Because the traffic relevant information gathered by SNMP is usually the throughput of a queried network device, such tools focus on traffic load monitoring and bandwidth usage classification. Because the data is retrieved at a constant time interval, it is considered as time series data and usually stored in the round-robin database (RRD). For such data structure, a straightforward and typical visualization is the line graph shown in Figure 1, which is generated by one of the most popular tool of the same kind named MRTG[9]. The vertical axis represents the traffic load while the horizontal axis represents the time line. Different types of traffic load can be plotted on the same graph and distinguished from each other by techniques such as color encoding, in order to visualize the bandwidth usage classification. In addition, the combination of the time series data

structure and RRD makes the visualization process very efficient regardless of the data amount. This is one of the most crucial reason that such system framework is very popular in practical large scale network administration.

NfSen uses netflow data instead of the SNMP data, but uses exactly the same visualization method as the one shown in Figure 1. Because it only focuses on the traffic load properties provided by the netflow data, namely Flows, Packets, and Bytes. Although NfSen provides many useful features such as filtering, aggregation and statistics by using other properties of netflow data as well, it gives up the opportunity to present new traffic characteristics other than load characteristics by ignoring the visualization of other properties than the three load relevant properties.

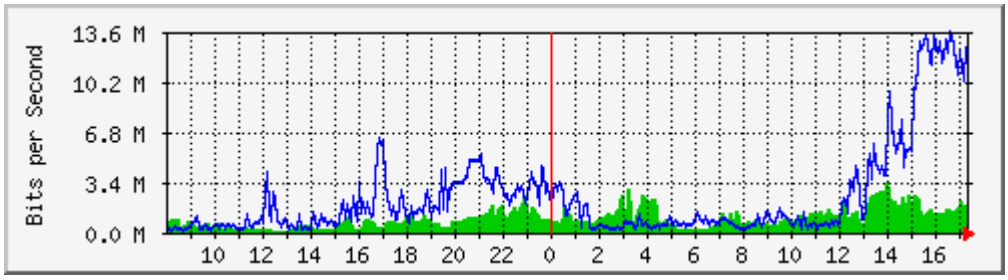


Figure 1: An example line graph produced by MRTG[9]

2.1.2 IDS Rainstorm and Rumint

Conti et al. propose in [10] the toolset composed of the IDS Rainstorm and the Rumint to confront the overload of security information, which is one of the most important motivations of this thesis work. As a matter of fact, this thesis starts from many inspiring ideas introduced in [10].

First of all, Conti et al. bring up the issue of information overload concerning security professionals. In short, the processing ability of human beings are exceeded by the enormous security information even though they are well trained professionals, as long as such information is still presented in the form of text. By bringing in the knowledge of human perception theory and cognitive science, they pointed out that such information overload can be solved or alleviated by replacing the text based presenting methods with the visualization based methods, due to the fact that the human visual perception system can be very efficient in perception of large amount of information as long as the information is presented visually in the right way. Such visual based methods distinguish themselves from the artificial intelligence methods by taking advantages of human brain directly rather than imitating the way it works.

Then, two prototype systems are given as a demonstration of concept. The first one is called the IDS Rainstorm, which initially appeared in [11], focused on alert logs generated by IDS systems such as Snort[12]. It uses a novel visualization combining the coordinate systems of scatter plot and parallel coordinate plot that are described later in Section 3.2. Specifically, as shown in

Figure 4, multiple parallel y-axes are all assigned to the IP address property in order to show the location of alarms, while a single x-axis is used as the time line to indicate when the alarms occur. Besides the main view, a zoom-in view is also provided when the user selects an area to zoom in. In the zoom-in view, two y-axes are preserved. The left y-axis represents the internal IP addresses (victims) while the right one represents the external IP addresses (attackers). The lines are drawn from the position of the external IP address on the right y-axis to the alarm point caused by it, so that the connection characteristic of the alarms is visually presented too. In addition, color encoding schemes are applied to reveal the severity and the amount of alarms.

The second tool Rumint focuses on network packets, which are captured by protocol and packet analysis tools such as the Ethereal and stored in the pcap format. Most of the visualizations here are about encoding packet byte into pixels. Besides the visualizations of packets in the binary level, which are very interesting but beyond our scope, we may learn from the combined visualization. As a combination of scatter plot and parallel coordinate plot again, this one differentiates from the one in IDS Rainstorm by separating the view port in three parts. A standard parallel coordinate plot with only two parallel y-axes stands in the middle, but the properties represented by each axis can be changed interactively. Two scatter plots staying on each side of the parallel coordinate plot share its two y-axes respectively. Both of their x-axes represent the time lines but have opposite directions. In this way, not only the relations between the selected two properties, but also the time patterns of those relations are depicted. Additionally, a VCR like animation over time is provided for all the implemented visualizations. Such playback assists the navigation of data and may give a clearer view of how network activity happens as time elapse.

Though Conti et al. in [13] summarize the interaction techniques applied in the two complementary tools as domain-specific semantic zooming, interactive encoding and dynamic querying, the lack of direct interaction on the visual objects leave the existing visualization with several problems which hinder potential further exploration into the data and reduce the analysis efficiency somehow. Such problems are further discussed in Section 2.2.3.

Lastly, a framework for the design of security visualization systems is also given based on the experiences gained in the implementation of the toolset. Such framework is described in Section 2.2.1.

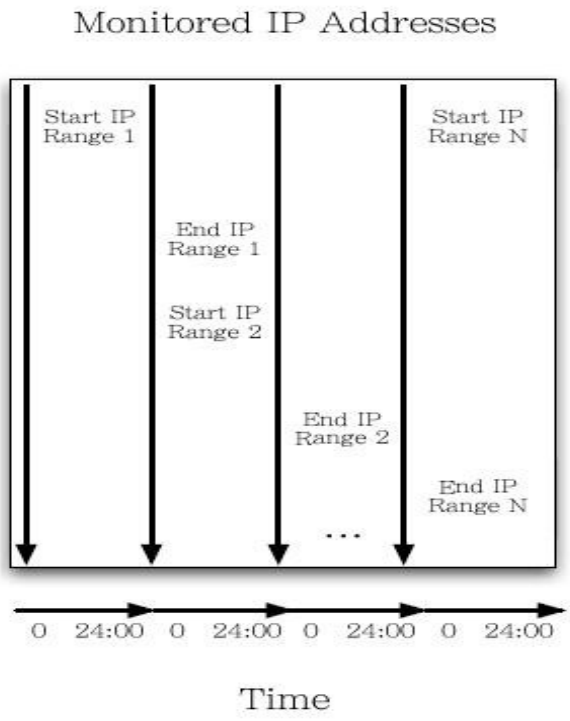


Figure 2: Plotting scheme of IDS Rainstorm[11]

2.1.3 SIFT

The SIFT[14] project proposed by William Yurcik et al. is a visual computer network security monitoring toolset designed directly regarding the practical operator interface requirements[15] for the netflow data. Similar to Conti et al.'s approach[10], SIFT consists of two complementary visualization tools as one for the holistic view and the other one focusing on a more specific view of the data, namely NVisionIP[16][15][17] and VisFlowConnect-IP[18][19][20] respectively. They directly take in Argus NetFlows[21] or NCSAUnified NetFlows as input data except for Cisco NetFlows. The data management tool CANINE within SIFT is used to convert between different netflow formats such as the Cisco NetFlows and Argus NetFlows. Unfortunately during our work, CANINE had not implemented the functionality of convertion from Cisco NetFlows to either of the other two NetFlow formats supported by the visualization tools yet.

The utmost requirement that NVisionIP attempts to fulfill is the ability to represent the state of all IP addresses of a large network in a single view. In addition, it should also allow drill-down to further details of the whole network on demand. Therefore it ends up with three views of different detail level, Galaxy view, small multiple view and machine view. The galaxy view is a scatter plot assign the horizontal axis with the subnet address and the vertical axis with the host address, whereby the data points is encoded to either color or shape according to the selected netflow property. Additional features include zooming and dynamic query. The other two detailed

views show statistical characteristics of certain netflow properties in histogram. By conveniently navigating between the above three levels of view, the relationships between the aggregated network activities and individual host activities should be identified and understood more easily.

The VisFlowConnect-IP simply adopts parallel coordinate plots to represent the connections between IP addresses. According to the operator interface requirements, its plotting scheme is set to three parallel vertical axes among which the middle one is assigned to source IP address whereas the other two on the sides are mapped to the destination IP address or subnet depending on the level of view. Thus each part of the view represents one direction of the flows. Symmetrical connections indicate normal established session between two hosts while asymmetry connections indicate scanning activities. Furthermore, by encoding the level of transparency or the thickness of the lines, the uplink/downlink proportion of the encoded property can be visualized as well. Such characteristics can be quite useful in security analysis.

2.1.4 VIAssist

D'Amico et al.'s VIAssist[22] is a visualization framework based on a comprehensive cognitive task analysis (CTA) of network security analysts. Though the CTA sounds a bit unfamiliar and is defined as “the study of an individual's or team's mental processing, activities, and communications within a specific work context”[22], it basically means the same as collecting firsthand requirements from the end users by modeling their best practices. Thus the resulting system meets the actual needs in practice with optimized operating efficiency.

In brief, the VIAssist framework consists of an in-depth event analysis view and a dashboard view of global activity, just as the complementary setup in the solutions introduced in Section 2.1.2 and Section 2.1.3. Based on this robust framework , VIAssist further distinguishes itself from other solutions by integrating several verified visualization techniques into the in-depth event analysis view such as histogram, parallel coordinate plot with glyph encoding. Commercial products such as Table Lens and StarTree from Inxight software[23] are also incorporated. These different in-depth views shares the same dataset so internal linkage is established by nature. In addition those visualization views are equipped with several intuitive visual interaction techniques for data manipulation, linked highlighting, and filtering. As one important result of the CTA, VIAssist offers some features to support collaboration among human analysts, mainly facilitated in a form of convenient documentation of the discoveries during the analysis. However, VIAssist is a commercial product rather than an open source project. This hindered it from being used in our work.

2.1.5 IDGraphs

Ren et al. build IDGraphs[24] in attempt to address four challenges in flow based IDS. These challenges are, identification of temporal characteristics of the intrusions in netflows observed from the edge network routers, identification of correlated novel attacks, interactive visual investigation, and appropriate threshold for automatic statistical IDS. Its solutions for the first three challenges are within our scope.

The first challenge is solved by using a plotting scheme based on scatter plot with the x-axis

representing the time series and the y-axis representing certain post-processed netflow property. Specifically such properties are the failed connection counts (SYN-SYN/ACK) for each aggregating key, which is the aggregation result of six carefully selected types of aggregation keys such as source IP address and destination port number. Such post-processed properties appear useful in suggesting scanning or TCP SYN flooding activities in which the connections are rarely established in general. In addition, the density of overlapped data points is mapped to the luminance, resulting in a contour map representing the density distribution of the data.

The second and third challenges are addressed by employing their Histographs visualization system to enhance the interaction capabilities. The interaction techniques includes interactive query by directly mouse clicking on the plotted data points, zooming on region selected by mouse dragging, and linked brushing of the selected data on different visualizations. Though the interaction features of Histographs seem very useful, the Histographs is not openly available and thus can not be utilized in our solution.

2.1.6 InetVis

InetVis[25] is a 3D scatter plot visualization tool adopted from the idea proposed in [26]. Unlike most other network visualizations choosing the lines to represent connections as a intuitive metaphor, the InetVis employs points for the same purpose. As an apparent consequence, the reduction of dimensions of the visual representation saves more display space and improves the rendering speed, at the cost of loosing the ability to separate the entities of each connection. However, such sacrifice does not necessarily result in the missing of connectivity characteristics, which is proved by the examples illustrated by the InetVis. Figure 3 explains the plotting scheme of InetVis very clearly by itself, except that the separate icmp plane below the cube turns out only 2 dimensional because the port number represented by the y-axis does not exist in icmp but in TCP or UDP.

Additional features of InetVis includes color encoding by selected traffic properties, resizing of data points, common manipulation of 3D space such as moving, rotating and zooming, and scalable playback animation.

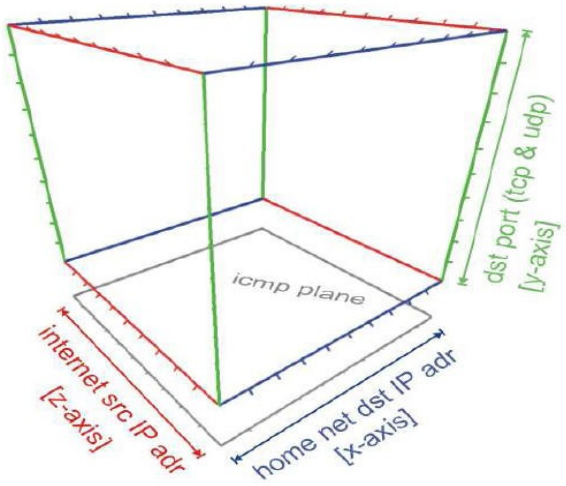


Figure 3: Plotting scheme of InetVis[25]

Making use of the plotting scheme in Figure 3 and additional features, InetVis is useful in identifying anomalies such as several scanning activities from the pcap data as demonstrated in [25].

2.1.7 AfterGlow

AfterGlow[27] is a script based visualization tool built for showing the relationships among the entities of the data input. It supports mainly two types of visualizations, namely network graph and treemap.

According to the project website, AfterGlow can be applied to various types of data sources such as network packet captures in pcap format, email logs, firewall logs, firewall rule-sets, web logs, IDS logs, OS logs, etc., but the main experience in its current usage is on network packet captures and email logs. It chooses a well known open source graph (network) visualization library named Graphviz[28] as its visualization back end, which is be very fast and scalable but does not directly support interaction.

Vorel et al. in [29] and Mansmann et al. in [30] give interesting examples of the botnet in large scale networks. The former uses network graphs to visualize the relationships between IP address and domain name hence the structures of the botnet stand out. The latter uses treemap to show the distribution of network autonomous systems (AS). As each square in the map represents an AS, its area indicates the number of IP addresses it holds and its color indicates the connection numbers within the AS. In this way the botnet propagation activities can be observed.

Solution	Data Source	Visualized Property	Visualization	Interaction	Characteristic
NfSen	Cisco NetFlow	Flow, Byte, Packet, Start Time	Time series graph	Filtering, Color Encoding	Throughput
IDS Rainstorm	Snort Log	IP Address, Severity of Alarm, Time stamp	Combined of Scatter Plot and Parallel Coordiante Plot	Semantic Zooming, Dynamic Querying, Color Encoding Animation	Connection, Alarm
Rumint	Pcap	All	Binary Rainfall, Scatter Plot, Parallel Coordinate Plot, Combined of Scatter Plot and Parallel Coordinate Plot	Multiple Views, Filtering, Color Encoding	Payload, Connection, Temporal,
SIFT	Argus NetFlow, NCSAUnified NetFlow	IP Address	Scatter Plot, Parallel Coordinate Plot, Histogram, Fish eye, Glyph	Dynamic Querying, Focus-Context Navigation, Zooming,	Connection, Statistics
VIAssist	NetFlow, etc.	All	Histogram, Parallel Coordinate Plot, Glyph, Table Lens, Network Graph	Dynamic Querying, Color Encoding, Multiple View, Linked Selection	Statistics, Connection,
IDGraphs	NetFlow	IP Address, Port, TCP Flag	Scatter Plot	Multiple View, Linked Selection, Luminance Encoding,	Failed Connection,
InetVis	NetFlow	IP Address, Port	3D-Scatter Plot	3D-Space Manipulation, Color Encoding	Connection
AfterGlow	Pcap, Many logs	All	Network Graph, Treemap	None	Connection

Table 2.1: Survey of typical existing VizSEC systems

2.2 Engineering the VizSEC System

This section summarizes the best practices in the development of the VizSEC systems. Such valuable experiences take a essential part in guiding the design of our VizSEC system in Chapter 5, together with the requirements from UNINETT[31] network security administrators as described in Section 4.4.

2.2.1 Design Guideline

Ben Shneiderman in [32] summarizes the basic principles from many visual design guidelines as the Visual Information Seeking Mantra: Overview first, zoom and filter, then details-on-demand. This is in accord with our survey results on the selected VizSEC systems in Section 2.1. Further more, there are seven tasks to accomplish[32]:

- **Overview:** Gain an overview of the entire collection.
- **Zoom:** Zoom in on items of interest.
- **Filter:** Filter out uninterested data.
- **Details-on-demand:** Select an item or group and get details when needed.
- **Relate:** View relationships among items.
- **History:** Keep a history of actions to support undo, replay, and progressive refinement.
- **Extract:** Allow extraction of sub-collections and of the query parameters.

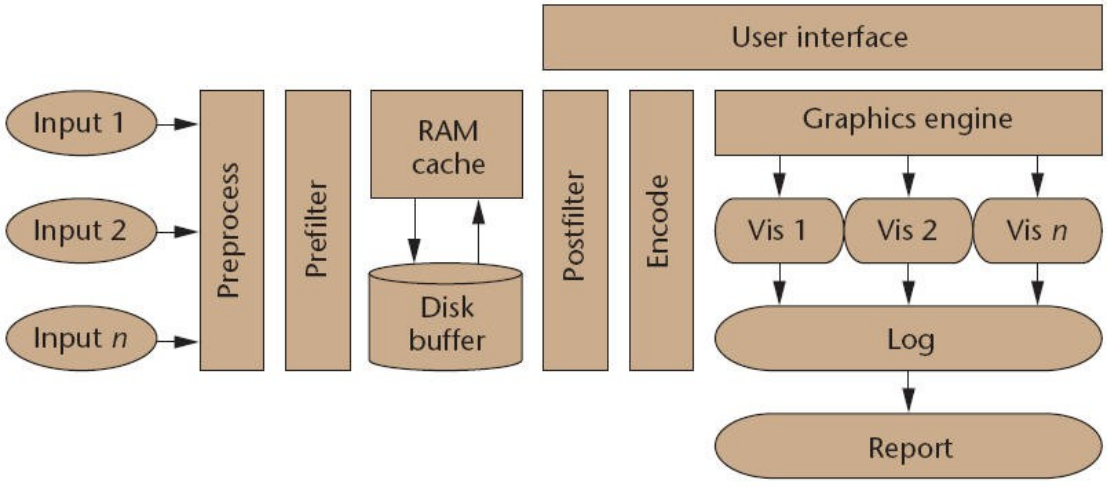


Figure 4: Design framework of security visualization system[10]

Meanwhile, Conti's VizSEC system design framework depicted in [10] fulfills the above principles and basic tasks perfectly. As shown in Figure 4, a VizSEC system should be able to take multiple input sources, to process and filter them before load them into RAM, and to swap them into disk buffer when necessary. Then, under the control of the user interface, the data loaded in the RAM or disk can be retrieved, filtered again, encoded according to specific scheme such as color, and finally visualized by the graphics engine. The resulting visualizations should be logged and further be compiled automatically into customized report.

2.2.2 User Requirements

The requirements of the network operator interface are summed up in [15] with the primary one as “the need for an overall situational awareness view of an entire network”. Some other

requirements in [15] which we also consider feasible to realize and important to be our criteria are listed as follows.

General requirements:

- User-friendly interface not requiring expertise knowledge
- Flexibility to query all properties of source data
- Dynamic view of network events over different time scales
- Cross-cueing between events
- Profiling of distinguishable classes of computers by activity type, activity volume, and time

Specific requirements:

- Raw port activity for well-known ports below 1024 and dynamic ports above 1024
- Indications of port activity above defined thresholds
- Drill-Down views of traffic by IP address
- Monitoring traffic exclusively to/from the Internet
- Monitoring traffic exclusively within the intranet
- Network mapping awareness
- Port scanning awareness

In order to enhance the discovery of novel anomalies, data mining techniques should be combined with information visualization[33]. As data mining relies on statistical algorithms and machine learning to find interesting pattern, allowing interactive input of human users to adjust the data mining algorithms according to their perception of the visualization should be a good approach towards the combination.

Last but not the least, the system should facilitate development and transfer of expertise and collaboration based on such expertise[34]. Logging of the visualization results and the procedures to generate such results can be a good start.

2.2.3 Defense of DoI Attacks

Though the VizSEC systems try to take advantages of the human brain in security analysis, there are also attacks targeting the VizSEC system as well as the perceptual cognitive and motor capabilities of human end users in order to diminish such advantages[35], namely the denial of information (DoI) attacks. Though Conti et al. in [35] describe exhaustively typical DoI attacks according to a DoI attack taxonomy by modeling the human processor as the targets, very few defense strategies are proposed. But the table 3.4 in [36] finally gives a technological DoI defense taxonomy which shows some sound directions. For example we can use filtering to reduce noise, optimize resource consumption to save limited resource, apply data fusion to reduce the amount of data input, and improve the human computer interface, etc.

3 Technological Background

This chapter presents the general technological background of NetFlow, information visualization and botnet. While NetFlow technology provides data input for the detection of malicious activities, visualizaiton techniques represent the NetFlow data source in the forms that can be efficiently comprehended by human users. Typical malicious network activities are then discussed in a botnet scenario.

3.1 NetFlow

In fact there are several NetFlow technologies that are slightly different from each other. In our case only the Cisco NetFlow technology is used.

3.1.1 Definition

Although the NetFlow is a Cisco patented technology, it evolves to a standard called IP Flow Information Export (IPFIX). The IPFIX standard defines a netflow as below.

> A set of ***IP packets*** passing an ***observation point*** in a network during a certain ***time interval***. All packets belonging to a particular flow have a set of ***common properties***.

The definition is further explaned in the following four aspects corresponding to the highlighted key words.

Observation Object

The basic object observed by NetFlow technology is the Internet Protocol (IP) packet. To get the values of the defined properties of the packet, the packet header of different protocols on multiple layers is taken and processed, whereas its payload is discarded. The values of certain properties (or attributes) are used as keys to distinguish each flow from others, while the values of other properties are aggregated to reflect the characteristics of each flow.

To reduce resource consumption, sampling mechanism is used in practice. When sampling is applied, only one packet out of every certain number of the observed packets is recorded whereas the others are discarded. Such “one out of N” scheme leads to a sample rate of 1:N. In our case the Cisco NetFlow provides two sampling modes, namely deterministic sampling and random sampling.

The deterministic sampling is a scheme in which the packets observed at the exact interval are taken. For example, at 1:100 sample rate, letting each packet be represented by its position in the arrival queue, the resulting packet sequence looks like “1, 101, 201, 301, ... “. This may miss most packets of a flow which arrives at a fixed pattern of interval. Thus the other scheme, i.e. the random sampling, is most practically used

The random sampling copes with the defect in the deterministic sampling by adding a random factor into the fixed sampling interval. In this way, the sample rate is statistically stable but the distance between sampled neighbor packets can still vary. For instance, at 1:100 sample rate, letting each packet be represented by its packet sequence number in the arrival sequence, the resulting packet sequence may look like “5, 120, 230, 302, ... “.

Observation Subject

As the observation is carried out on the network layer, it is no surprise to see that layer-3 network devices such as routers are in the right position to serve as the observation points. Of course, such router must realize the NetFlow technology and enable the exporting function in order to act as an NetFlow exporter in Figure 6.

Measures

NetFlow-enabled router maintains a local cache to generate flow records as measures of the traffics. When a new packet arrives, the router first determines whether it belongs to any of the flow entry recorded in the cache by matching their values of certain properties called flow keys in this work.

The flow key is composed by seven properties to uniquely identify each flow, as depicted in Figure 5.

If the packet does not belong to any existing flows, a new flow entry containing the values of certain properties of the packet is created.

If the packet belongs to a flow entry in the cache, the router update the measurement data for this flow, for example, the accumulated traffic amount in byte of this flow.

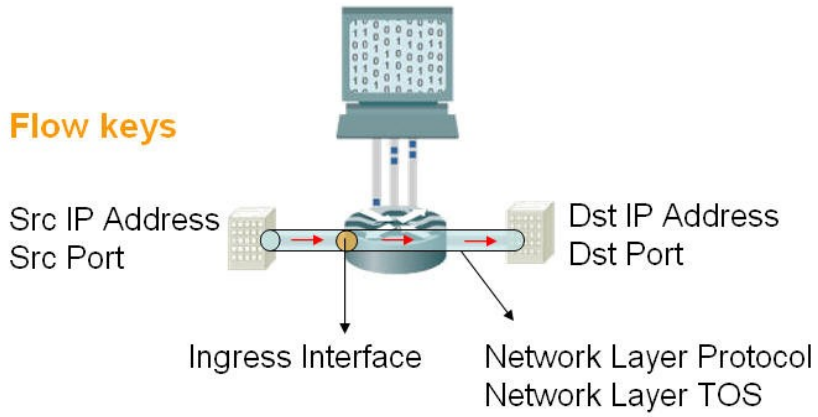


Figure 5: Flow keys

On both sides of the flow, the IP addresses in the IP header and the port number in the header of the transport layer protocol such as TCP and UDP are marked as source or destination of the flow according to the full field name of the IP address in the IP header. Because the source and

destination IP addresses are both used as flow keys and treated as different properties, the flow defined under this restriction is obviously unidirectional.

The IP address tells the participant of the flow. The port number indicates the application carried on this flow, or set to zero if not available. IP address and port number are reckoned suitable properties to represent the connection characteristics of the flow.

When the flow runs through the observing router in the network, the local number of the ingress interface on the router is chosen as another flow key, because a router normally have more than one ingress interface. By combining with the routing table of the router, the knowledge of the ingress interface can be used to identify the next hop of this flow.

The other two flow keys are the type of service and the type of protocol on the network layer of the OSI model. They are useful in basic traffic classification in general network monitoring and planning but do not offer much help in security analysis.

In addition to the seven flow keys, some other flow properties are also recorded. Here we mainly describe the properties that we use in our implementation and explain why the rest properties are ignored.

Time-strart, time-end and duration are all temporal properties of the flow. They are very important in security analysis because all malicious activities would more or less have specific temporal characteristics.

TCP flag could be a helpful property in identifying DoS attacks based on the three-way handshake procedure to establish TCP connection. However, the nature that the value of this property is the combination of all observed TCP flags in the flow diminish its effectiveness in the identification of such anomalies. Surely this property is useless when the TCP protocol is not used in the flow.

Last but not the least category of useful properties in security analysis is the payload statistics including total byte number, total packet number, and byte per packet. As the payload is discarded in netflow records, such statistical properties are the only clues to tell the characteristics of the payload.

As for the rest of the flow properties, which are less useful in direct visualization in our specific environment, are not used in the implementation in order to save limited resources. However, they should be easily added if required by the users. In addition, they should be also useful in the preprocesses of the data before visualization, such as aggregation and filtering.

The source and destination autonomous system (AS) of the flow may be good to show the distribution of the other properties over a network-wide view, which is coarser and higher than the host wide view based on IP address. However, our targeting observation points are mainly routers located in the backbone network of a large ISP and very little variety of AS can be observed.

Though the ingress interface is one of the flow keys, the ingress router interfaces of a flow does not provide much information about the characteristics of the traffic itself, especially when the routers are deployed within a core network in which the routing between routers are static. Putting the NetFlow records from multiple routers all together may give some interesting

characteristics, but is out of the scope of our work.

Temporal Setup

All computer systems have limited resources and must have mechanism to balance the resource consumption so that the system can work continuously with a smooth performance rather than chock up or even halt.

Normally, when a packet indicating the end of a flow, such as the a TCP packet with the FIN flag flipped on, NetFlow routers can understand and mark the corresponding flow entry as finished and ready to be exported with other finished flows. In other words this flow entry is frozen until being exported and cleared from the cache.

As a matter of fact, NetFlow introduces two timeout counters for each flow entry. One is idle counter that resets to zero and starts to increase along time after the last arrival of packet. Therefore, the counter indicates the inactive duration when no packet is observed in the flow. Once reaching the threshold the flow entry is marked as finished

The other one is active counter which starts and keeps increasing along time since the creation of the flow, as long as the relevant flow entry is not marked as finished. Therefore this counter indicates the total duration of the flow. Once reaching the threshold, the flow entry is marked as finished, even though the actual transmission within the flow is still active. Newly observed packets share the same values of the flow keys with the active timed out flow are recorded in a new flow entry and thus considered as a new flow. Lower active timeout threshold produce more fine-grained flow observations, at the cost of the probability to tell the real duration of the flows.

3.1.2 Architecture

NetFlow technology is a system consisting of three logical components: NetFlow Exporter, NetFlow Collector, and NetFlow Analyzer. Figure 6 shows the relationships between each component and their functionalities.

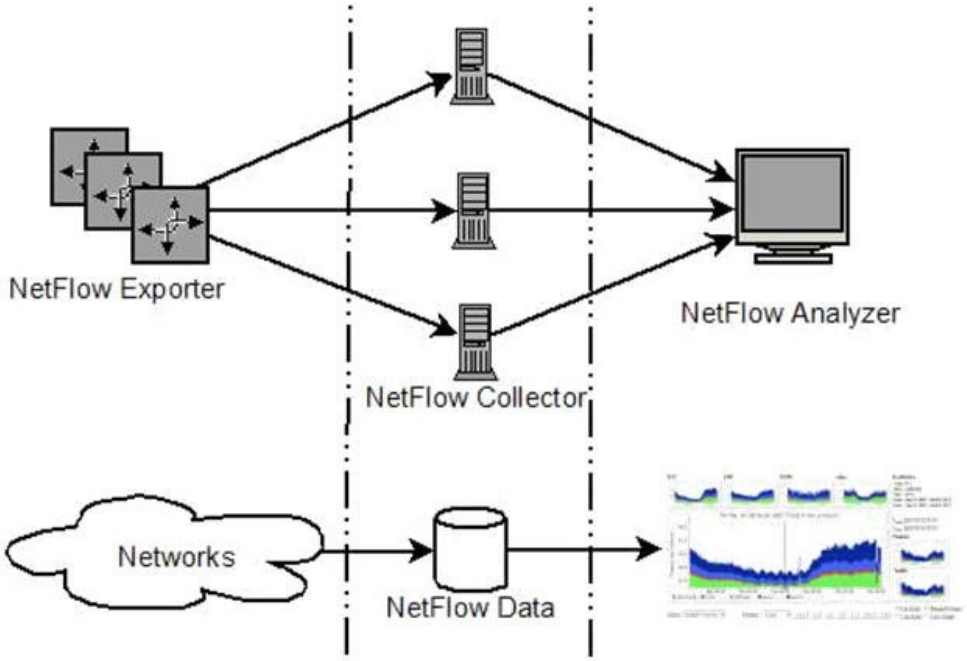


Figure 6: NetFlow architecture

As mentioned in the previous section, a NetFlow exporter is a NetFlow-technology-enabled router deployed at a certain observation point in the network. It captures bypassing packets, classifies them into flow entries in its local cache, and exports the finished flow entries to NetFlow collectors.

A NetFlow collector is usually a UNIX server equipped with a NetFlow collection software, such as the nfcap in the NfDump toolset in our case. Such software provides a daemon service listening on the network. Whenever the flow records exported by the exporter arrive at the collector, such daemon programme receives the records and stores them onto the hard disk, normally in binary format for storage size optimization.

A NetFlow analyzer is usually a terminal with NetFlow analysis software installed. Such software converts the binary raw flow data into human readable format, hence should probably in the same tool suite with the collecting software mentioned above in order to recognize the binary format. The software also provides analysis functionalities with human-computer interface that allows the human user to make use of the data. In our case NfDump includes nfdump for format conversion and several analyzing functionalities such as filtering, aggregation, and top talker statistics. More sophisticated software such as NfSen described in Section 2.1.1would be very good to have on the analyzer too in order to help the human users gain more insights into the data and to enhance the analyzing efficiency.

Because the collector and the analyzer are mainly determined by the software installed on them, their functionalities can also be fulfilled by a single computer with both softwares installed.

3.2 Information Visualization

The term information visualization here covers a much wider range of domains than visualization techniques. As a general and comprehensive description of information visualization, this section explains the most relevant knowledge of those domains. Given not only how to visualize but also why the visualization works, we can gain more in-depth evaluation of the design and implementation of our VizSEC system. The main reference of this section is Colin Ware's book “Information Visualization: Perception for Design”[5], in which many principles and techniques have been applied and proved in the VizSEC systems surveyed in Section 2.1.

3.2.1 Human Perception and Visualization

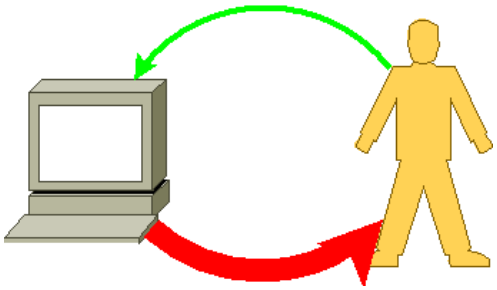


Figure 7: Human-computer problem solving loop

In general, human beings have very special perception capabilities which are way beyond the performance of any of the state-of-the-art AI algorithms in discovering patterns. On the other hand, the major advantage of computers over human beings is its high speed in executing pre-defined computational tasks. Combining the discovery capability of human beings and the fast computing capability of computers, we have a problem solving system with a loop as shown in Figure 7, wherein the thick red arrow indicates the abundant information produced by the computer and the thin green arrow refers to the analysis result that is given by the human as a guidance for the computer to produce information for the next problem solving cycle if the problem is not solved yet.

In our scenario, the computer generates visual images of the netflow data for the human user to discover malicious activities hidden in the data. In each analysis cycle the human user gives feedbacks to the computer in order to get more desirable visual images that lead to a step forward to the objective. Obviously, visualization acts as the interface between the two components. In order to optimize the efficiency of the system, more detailed look into the loop is necessary.

Reference Model of Visualization

As depicted in Figure 8, the processing flow of computers and the perceptual flow of human beings are enclosed in the two boxes, illustrating a reference model of the visualization.

The processing flow of computers in the left box looks very straightforward. Starting from the raw data, the computer first dumps the raw data to the data table. Each row of the table represents a data record, whereas each column represents a property of the data record. Any data that can be

converted to such form of data table is called table data, so does the NetFlow data in our case. The data table can also be acquired indirectly by applying a pre-processing on the raw data. For example, linear transformations such as fast Fourier transform (FFT), dimension reduction methods such as principle component analysis (PCA), clustering algorithms such as multidimensional scaling (MDS), or other semantic procedures that convert the raw data into data tables is considered valid prior model.

Then, the records in the data table are mapped visually to some visual structures such as points, lines, areas, etc., and finally rendered to views displayed in the output device such as a monitor. The process between data tables and views is visualization.

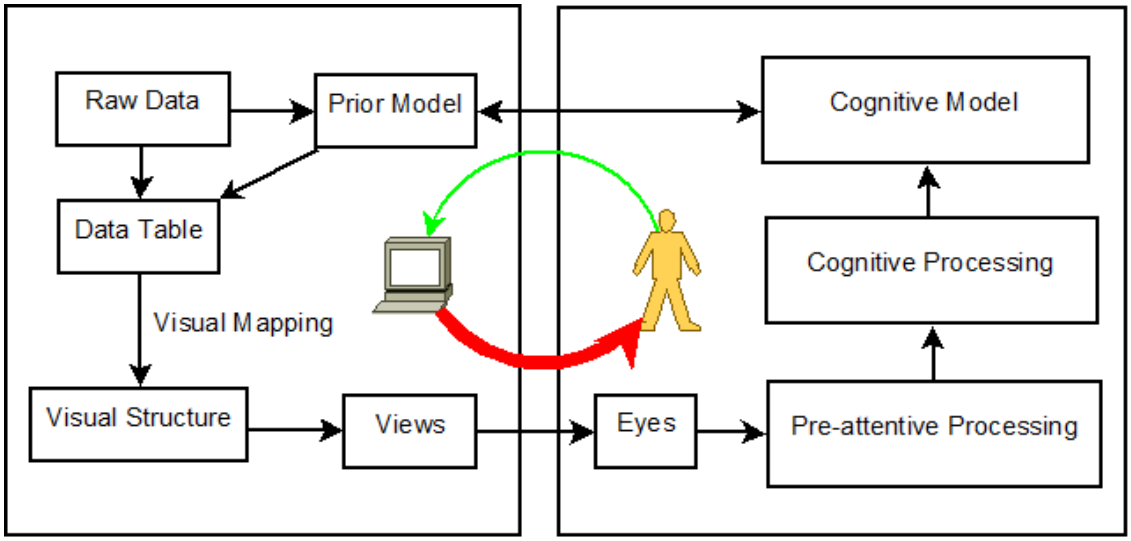


Figure 8: Detailed problem solving loop with the reference model of visualization

The perceptual flow of human beings in the right box can be divided to two parts: the vision input by the sight of eyes, and the mental perceptual processing of brains.

Apparently, as the eyes are the only interface to take in the views, the faithfulness of the the conversion from the real world view to mental view becomes critical. However, human eyes can never take all information in the real world, because of many natural limitations. For example, the limit in acuity, contrast sensitivity, color sensitivity, and so on. Besides, certain pattern such as high frequency strips may easily cause visual stress, which significantly restrict the continuous working period of eyes. If the views are carefully designed with respect to such limitations, the performance can be optimized.

After the first step, the human brain takes over further perception according to the reference model of visualization. This consists of three stages of perceptual processing as follows:

1. Pre-attentive processing

   This is the first processing that extract low-level properties of the view captured by eyes. As a result of evolution, the eye can perform very fast processing to extract certain features in parallel, and stored them in the visual working memory. These features are so called pre-attentive features, which are exemplified in Figure 10. Symbols of such

features are called sensory features. More details on working memory are given in Section 3.2.3.

Because this processing is done without consciousness, it is a bottom-up data driven processing. The speed of pre-attentive processing is independent of the number of distracting objects, while the processing speed of non-pre-attentive features is slower and the speed decreases as the number of distracting objects increases.

2. Cognitive processing

   Also called pattern perception, this processing is slow serial processing involving both working memory and long-term memory, which are further discussed in Section 3.2.3. Compared to the sensory symbols that the pre-attentive processing works on, cognitive processing perceives arbitrary symbols that require cognitive efforts recalling knowledge in the long-term memory. As presented in Figure 9, it would be much more difficult to relate the Arabic symbol “3” in the left big circle to the meaning of “three”, than to relate the three small circles in the right circle to the same meaning, unless the Arabic symbol is learned in prior.

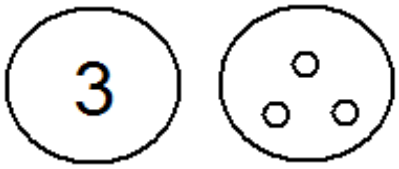


Figure 9: Arbitrary symbol(left) vs sensory symbol(right)

3. Goal-driven processing

   Using the objects recognized in the cognitive processing, this stage of processing attempts to form cognitive model for certain goals. Requiring high level reasoning in a sequential manner, it is the slowest processing in the whole perceptual flow. The resulting model or pattern is then sent to the computer to influence the next visualization processing.

Pre-attentive Features

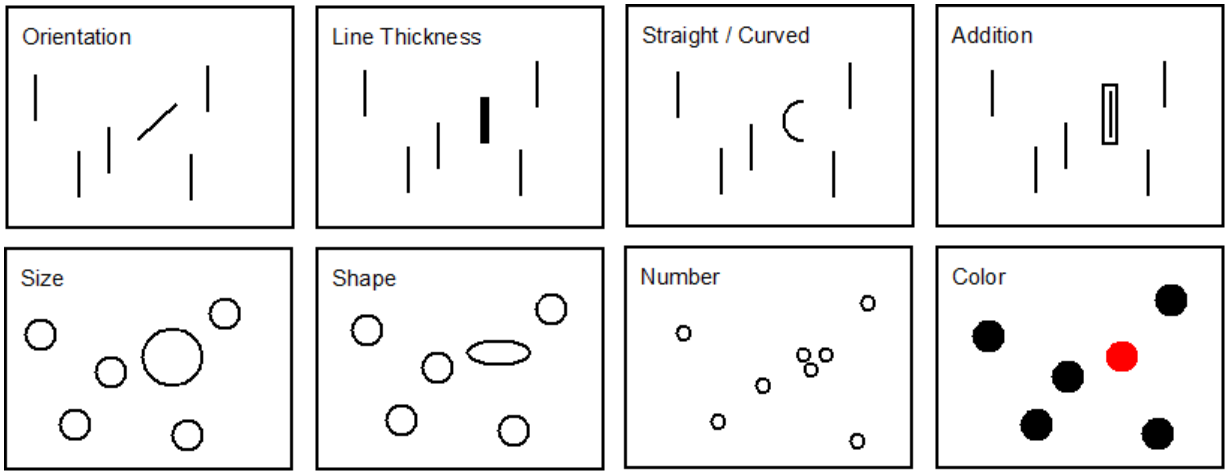


Figure 10: Pre-attentive features

As mentioned before, pre-attentive features are certain forms that can be rapidly perceived in parallel without conscious attention. Some examples of pre-attentive features are presented in Figure 10. Such features should be encoded to the data representation in order to enable fast identification of special visual patterns.

In addition, Gestalt (means “form” in German) laws provide several empirical guidelines to use pre-attentive features to achieve visual clustering. For instance, objects that are close, similar, or vertically symmetrical to each other can be pre-attentively perceived as a group. This is very useful in fast visual classification of views containing enormous data points.

3.2.2 Plotting Schemes

In Section 3.2.1 we explained that netflow data is table data. Because table data have the same multiple attributes for every data record, such data is also called Multidimensional Multivariate (MDMV) data, where multidimensional refers to the dimensionality of independent properties and multivariate refers to the dimensionality of dependent properties.

The visualization of such MDMV data[37] is not intuitive but several techniques are proved to be effective and widely adopted, such as line graphs, scatter plots, histograms, glyphs and icons, recursive visualizations, clustering visualizations, pixel-oriented techniques, and so on. In this work we choose the simplest and most popular ones according to the design principles in Section 5.2.

Scatter Plot

Scatter plot is the most popular visual data mining tool that is good at finding outliers and observing clusters and correlations. Though the plotting scheme can only represent two dimensions of data, by employing additional techniques such as multiple views, glyphs and icons, color encoding and so on, scatter plots can definitely represent higher dimensions of data. Figure 11 illustrates the plotting scheme of scatter plot.

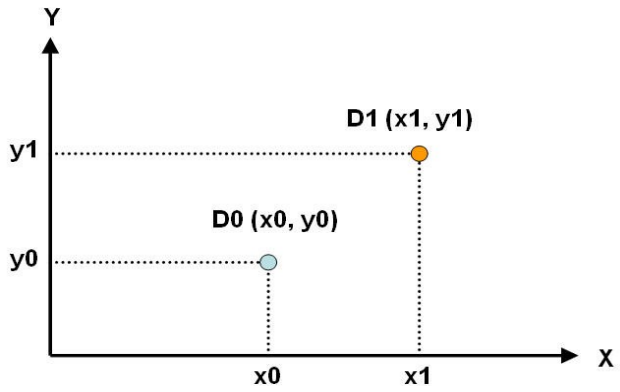


Figure 11: Plotting scheme of scatter plot

Histogram

As presented in Figure 12, histograms are bar charts in which the height for the bar represents the total amount of data points falling in the value range on the x-axis. This value range is determined by the positions at which the two edges of the bar located on the x-axis.

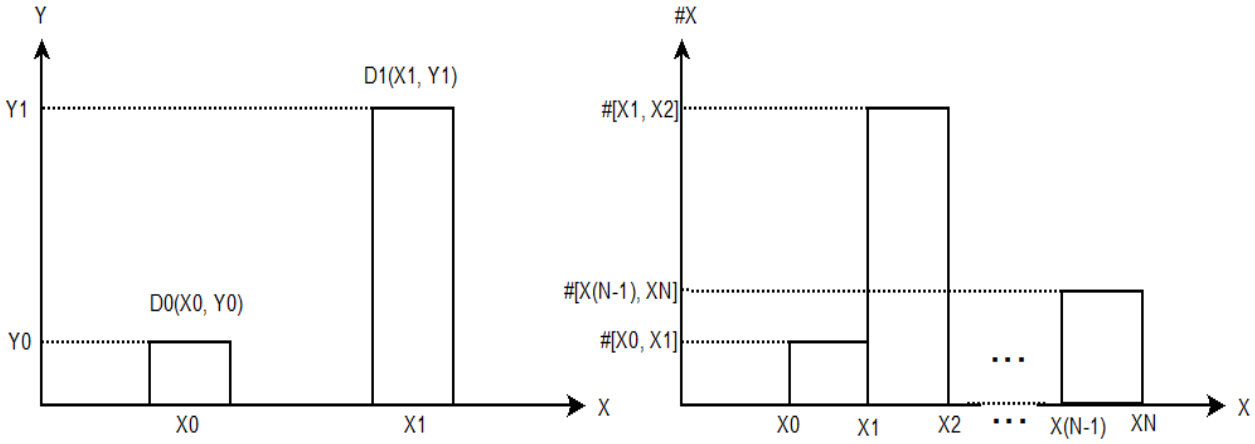


Figure 12: Plotting scheme of bar chart(left) and histogram(right)

Though its plotting scheme only supports direct representation of data in one dimension, by using the same techniques afore mentioned, histograms can be used to represent higher dimensions of data too.

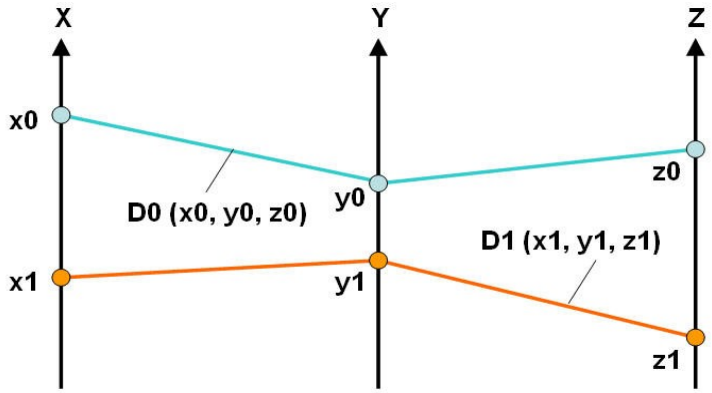


Figure 13: Plotting scheme of parallel coordinate plot

Parallel Coordinate Plot

Parallel coordinate plot (PCP) is mostly used as a visual method of the clustering visualization techniques. Unlike the scatter plots and histograms that need additional techniques to show the data in more dimensions, PCP can supports to very high dimensionality by simply increasing the parallel y-axes as shown in Figure 13.

3.2.3 Interaction Techniques

In Section 2.2.1, we learned from the survey result that interaction functionalities are indispensable requirement for a good VizSEC system. According to the problem solving loop and the reference model of visualization in Section 3.2.1, we understand that visualization is the interface that closes up the loop and interactive visualization is the most direct way to enhance such interface. Furthermore in Section 3.2.2, to deal with the MDMV data, most of our selected plotting schemes must add extra interaction features. All these show that interactive visualization must be carefully designed in our solution.

Focus+Context Problem

The first issue that interaction techniques need to address is the focus+context problem in navigating the data views. This is a challenge when confronting large scale data sets like the netflow data.

More specifically, the focus+context problem is about how to find details from a larger context in given information space. Several techniques are proved effective in addressing such issue, such as the elision techniques, in which part of the structure is hidden until needed; or the distortion techniques, in which regions of interest are magnified while the space of irrelevant regions are compressed; or rapid zooming techniques, by which users can conveniently zoom in and out of interested regions; or multiple views, among which some views give overview and others focus on details.We choose the rapid zooming and the multiple views techniques again because of their simplicity and verified efficiency. To enhance the relationships between different views, we

feature our prototype system with linked brushing technique as presented in Figure 14. This technique enables automated highlighting of selected data subset in all views.

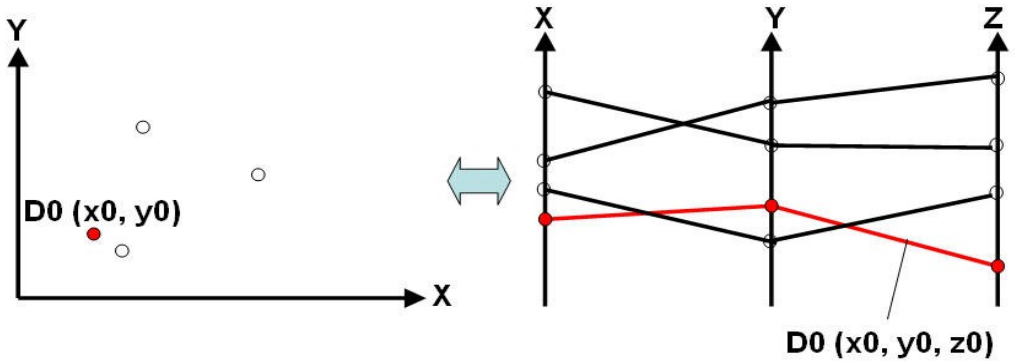


Figure 14: Linked brushing between scatter plot (left) and parallel coordinate plot

Interactive Visualization Feedback Loops

In general, there are three types of feedback loops in the interactive visualization:

1. Visual-manual control loop for data manipulation, where human users act on seeing visual interface components such as buttons;
2. View refinement and navigation control loop for exploratory navigation, which corresponds to the pre-attentive processing and the cognitive processing;
3. Problem solving loop for the discovery of cognitive models and patterns, corresponding to the goal-driven processing.

Relevant time scales corresponding to each class are:

1. 0.1 second correspond to very fast psychological moment;
2. 1 second corresponds to slightly slower than previous one because cognitive effort is needed to response to unprepared situation;
3. 10 seconds correspond to the slowest sequential reasoning of unit tasks.

Besides the reference model for visualization, the visual working memory and visual attention theory can also explain the above time scales and probably shed light on the optimized utilization of the interaction techniques.

In brief, the human perception system has a temporal memory similar to the random access memory (RAM) in common computer architecture. Such memory is called working memory which is supposed to extremely fast. But it has very small capacity and is only for the short term caching. The visual working memory of a common human can hold only three to seven visual objects and patterns, which stay at most a couple of minutes. Putting content into the visual working memory requires visual attentions, whereas storage time longer than 3 seconds requires cognitive effort. Even worse, performing cognitive effort rapidly used up the bandwidth available for other tasks that ask for working memory.

According to the visual working memory theory, the interaction techniques should try to reduce the total number of views required in a perception task. Also, each small perception sub-task forming a complex task unit should be as simple as possible in order to finish the perception as soon as possible. Hence, the cognitive effort can be reduced or avoided so that other small task can use the working memory as well to perform efficiently. In addition, the VizSEC system should be fast enough in providing views within the time scale of each type of loop, otherwise the working memory will be forced to perform cognitive effort in order to store certain content for a task for a longer period. During this period, other task can not efficiently utilize the working memory because of the limited bandwidth.

3.2.4 Multidimensional Reduction

Strictly speaking, multidimensional reduction is not mainly for information visualization. We discuss about here along with other visualization issues because it provides another perspective of MDMV data analysis than direct visualization methods.

Multidimensional reduction methods attempt to project the data from a high dimensional space to a lower dimensional space but also to preserve the characteristics of the original data at the same time. In our case, some netflow properties would be chosen first in order to determine which columns of the data table should be processed. Because we want to visualize the new data of the newly produced properties mainly by our two-dimensional scatter plot, we usually set the number of the outputting lower dimension to two accordingly.

As for the reduction methods, the principle component analysis (PCA) should be the first option. This is due to three reasons comparing to other methods. First, the meaning of PCA is relatively intuitive. It tries to find an orientation in the high dimensional space along which the data should be projected, so that the resulting data points projected to the lower dimensional space preserve the greatest variance. Because some data points are inevitably collapsed after the projection, PCA is usually more continuity optimized than trustworthiness optimized. Second, the algorithm for PCA is systematic and could be very efficiently performed by computer. It is considered one of the fastest multidimensional reduction methods. Third, PCA has been used for over a century since its invention in 1901 by Karl Pearson. As one of the most popularly used exploratory data analysis methods, it is proved reliable to find the factors of the data under three main assumptions. First, the data is linear composition of certain basis. Second, the data is Gaussian data in which statistical characteristics such as mean and covariance are important to depict the data. Third, large variances have important dynamics. This assumption is mainly for the interpretation of the result of PCA and based on another assumption that the data have high signal-to-noise ratio. Hence the resulting principle components are considered as useful signals rather than useless noise.

After the original data have been mapped to a lower dimensional subspace, plotting schemes designed for low dimensional data could be used directly on the resulting data. However, because the difficulty in interpretation of the result of multidimensional reduction alone, visualization of the original properties should be used in combination with the new properties.

3.3 Botnet

A botnet is a network mainly consisting of compromised computers called bots, which are controlled by other computers within the network. Those controlling computers use certain network communication links to send commands to the bots. Thus such controlling computers are called command and control (C&C) servers or simply controllers in this work, while the communication links are called C&C links. Because controllers are in direct contact with the bots, they may expose themselves too much and get caught for their illegal behavior. Thus normally, the controllers are also compromised computers instead of the real bot herder. Figure 15 shows a basic botnet architecture, where the controller can be more and even organized in several layers in order to hide the bot herder better.

Characterized by stealthy, organized and profit-driven, botnets are capable of performing various malicious tasks from sending spam to operating large phishing system.

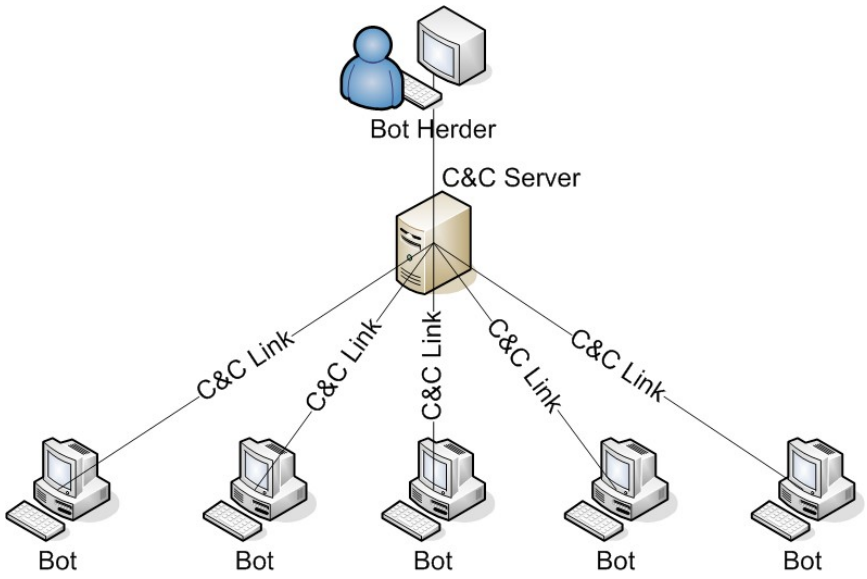


Figure 15: Basic botnet architecture

We focus on botnets for the detection of malicious activity for two main reasons. First, botnets as a collection of the organized compromised computers in the networks, perform almost all typical kinds of malicious activities. Hence by looking into botnets, we understand major malicious network activities as well. Second, according to the latest worldwide infrastructure security report[38] by the Arbor Networks, which covers 12 months from July 2006 to June 2007 and focuses on operational network security issues, botnets have become the most significant security threats together with distributed denial of service (DdoS) attacks in views of worldwide ISPs.

Just like any criminal organization, any malicious task performed by botnets can be separated to four stages: reconnaissance, propagation, communication, and finally attack. Each stage involves certain malicious activities, such as scanning for reconnaissance, spam relay and malware distribution for propagation, C&C link for communication, and finally causing DoS [39] for attack. Though botnets can carry out many other malicious activities such as click fraud,

phishing, brute-force password guessing and so on, we only focus on the activities discussed as follows for two reasons. One reason is that their instinct behavioral characteristics will probably result in strong patterns in netflow data to be discovered. The other reason is that they are the most prominent malicious activities across the Internet according to the observation of ISPs[38].

- **Scanning**

  Scanning is the activity that scanners (computers connected to the networks) try to find potential vulnerable computers that could be further exploited on the networks. Usually the number of scanners are much smaller than the number of victims for cost-efficiency reasons, and the time duration is short.

  In general there are two basic types of scanning. One targets IP address and the other one targets port number in TCP or UDP headers, namely IP-sweep scanning and port-sweep scanning. IP address based scanning attempts to find vulnerable computers within a network block, thus they result in traffic flows with very few different source IP addresses but a lot of different destination IP addresses within a network block.

  Meanwhile, port number based scannings attempt to find vulnerable services opened on targeted computers. They leave main traces in the traffic flows characterized by very few different source IP addresses but many different destination port numbers. In addition, other properties of the flows are usually the same or very similar.

  Most of the time the scannings are carried out in a very short time period, such as a few seconds or minutes. When the period last much more than that, they become very difficult to be discovered and are named creepy-crawly scannings.

- **DoS Attack**

  DoS attack is the activity that attackers (computers connected to the networks) try to exhaust computer resources or network bandwidth, in order to prevent them from providing services to legitimate users.

  The resulting effect of DoS attack mainly depends on its impact on the target and its similarity to legitimate traffics. In other words, it generally cumulates as many attackers as possible to send a large enough amount of legitimate service requests to the targets per time unit in order to exhaust their capacities. Thus, effective DoS attacks usually lead to traffic flows with a lot of different source IP addresses but only very few different destination IP addresses. Then they are called Distributed DoS (DDoS) attacks. In addition, like scanning, other properties of the flows are usually the same or very similar.

  Two major types of DoS attacks are TCP SYN flooding attacks in which the packets have the SYN bit in the TCP header set in order to exhaust the TCP connection capacity of the targets, and UDP or ICMP flooding attacks in which the packets are very large in order to exhaust the processing resources of the targets[38]. Such DoS attacks are brute-force attacks which are simple but effective.

- **C&C Link**

  C&C links are not such harmful threats by themselves unless they are exploited to organize the bots to perform malicious activities. Via the C&C links, hundreds of

thousands of bots formed a powerful army under the control of bot herders. Of course, on the other side the bots can do nothing without such C&C links. In addition, the C&C link is the key to trace back to both controllers and bots, which is the first step to clean up the infected computers in order to diminish the scale of botnets and mitigate the effect of their attacks. Therefore, C&C links are considered the weakest part of botnets. In the mean time, stealthiness is the most essential characteristic that all C&C links try to improve on.

In order to hide from detections, botnets use legitimate protocols such as IRC and HTTP for their C&C links. Usually a botnet is denoted by the protocol used in its C&C link, such as IRC botnet, and HTTP botnet.

In the IRC botnets, the traffics of the its C&C link are very similar to normal IRC traffics. As a instinct behavior of IRC protocol, the controllers need to send very frequent periodical keep-active signals to their bots in order to maintain the aliveness of the C&C links connecting them. Thus once a controller is identified, it is quite easy to find the bots under its control via the keep-alive signals. Moreover, the “push” style of dispatching controlling commands in IRC channels exposes the malicious intention of controllers.

In the HTTP botnets, by using the “POST” method in the HTTP protocol, the bots take the initiative to “pull” the controlling commands from their controllers. The “pull” style hides the C&C links better than the “push” style by allowing more flexible temporal communication patterns for different bots.

Fortunately, the botnets based on IRC C&C links are still the majority of current botnets mainly due to their simplicity. However, emerging botnets such as HTTP botnets or even peer-to-peer (P2P) botnets are more difficult to be detected, hence are more harmful and deserve more attentions in the efforts for their detection and mitigation.

Most of the time, the C&C link is used for the following same tasks. First, the bots need it to update their malware in order to perform specific malicious activities. Second, the controllers need it as keep-alive signal to keep in touch with their bots. Third, the controllers need it to send specific commands to their bots in order to launch destined malicious activities. The first two tasks are very critical because if we can detect C&C links by them, we can prevent or at least mitigate further destructions caused by the last task by cleaning up the bots and quarantining the controllers in advance.

4 UNINETT CERT

UNINETT[31] deploys and maintains a national research based computer network with high speed connectivity in Norway. UNINETT CERT is the security team in UNINETT that is responsible for monitoring the network, alerting security events, and coordinating other relevant departments to respond when security incidents occur. This thesis work fits itself in the proactive part of UNINETT CERT, which concentrates on active prevention and detection of malicious network activities.

4.1 Motivation

From such a big network connecting most educational and research institutes in Norway with high speed connectivity, UNINETT collects more than 30 GB NetFlow data per day from its 27 Cisco NetFlow-enabled routers with 207 interfaces at the sampling rate of 1:100.

The same as many ISPs in the world, UNINETT's major concern of malicious activities is botnet activities. Using its own NetFlow data source, the UNINETT CERT intends to find the bots and clean them up in order to mitigate the botnets from their source of power.

4.2 Facility and Resource

In front of such massive amount of data, the small scale of the CERT is a too limited resource for their challenging. However, the work has to be carried out in a way.

As a result, UNINETT CERT takes the “picking the low-hanging fruit” principle to guide their work. In other word, they try to dedicate their limited resources in the very tasks that bring good result fast.

For example, they have set up and maintained honeypots to study the most up-to-date malwares. By performing code analysis on the malwares, useful information about the botnets is revealed, such as the IP addresses of the controllers, and the programmatic behaviors of the bots that execute such malware.

Another “low-hanging fruit” picked is the darknet facility. A darknet is a routed allocated IP space in which no active services reside. Obviously, assumption can be made that all traffics flow into the darknet are abnormal traffics.

Last but not the least, based on a lively up-to-date list of the IP addresses of the controllers, UNINETT CERT routes the traffics destined for the known controllers to a sinkhole host, where the malicious traffics are analyzed and dropped in the end. This facility is actually very effective in mitigating the botnet relevant malicious activities by directly cutting off the C&C links.

4.3 Existing Solution

In order to utilize the enormous NetFlow log data better, UNINETT CERT attempts to explore

the data directly. However, the existing solution, namely the Stager system, is based on the NfSen project. As introduced in Section 2.1.1, the NfSen focuses on the analysis of the traffic load characteristics using line graph visualizations and other features such as filtering and aggregation. This solution appear insufficient and clumsy in security incident investigation so far, because of its neglect of other traffic characteristics than the load characteristics and its limited interactive visualization supports.

4.4 Wishing List

According to the situation described above, UNINETT CERT makes the following wishing list for this thesis work:

- Build a tool to visualize more properties than load properties of the NetFlow data.
- The tool should be scalable enough concerning big data amount.
- The tool should support convenient navigation between overview and detailed view.
- The tool should support detail-on-demand.
- The tool should keep trace of investigation procedures, and allow the application of the same procedure to different datasets.
- The tool should support multiple investigation on the same data, and provide means to establish relationships between them.
- Establish relationships between actual activities and basic visual patterns observed from views produced by the tool.

Generally, this list is a subset of the requirements given in Section 2.2.

5 System Design

Bearing Shneiderman's Visual Information Seeking Mantra in mind, we design our own VizSEC system – UniVis, mainly by filling concrete components into Conti's design framework as described in Section 2.2.1.

In order to meet the general user requirements summarized in Section 2.2.2 with special emphasis on the wishing list of UNINETT CERT in Section 4.4, we try to choose the most suitable components to be incorporated into our system. Our selection is mainly based on the survey results in Section 2.1 which reveal verified advantages and disadvantages of each existing VizSEC system. In addition to the preliminary empirical knowledges, several implementation principles concerning our specific using scenario are also employed in the selection.

5.1 Requirements

5.1.1 Global Overview Snapshot

The first requirement of VizSEC systems shared by security administrators is the ability to provide the overview of an entire network. In our case, the netflow data collected from a backbone network router is so enormous that even the data of one hour is probably too much for interactive operations by our simplest and thus most efficient visualization scheme. Therefore, we suggest an auto-snapshot solution. By auto-snapshot, we mean a predefined schedule following which the system automatically visualizes the specified dataset in specified ways without interactive operations of human users. The views generated are saved as image files for off-line routine analysis.

5.1.2 Local Details Drill-down

The second requirement is the interactive visual exploration ability that allows further drill-down into the interesting local spots of the overview. The relevant features are zooming, filtering, detail-on-demand, etc. Zooming provides clearer view of local area. Filtering excludes unwanted data. Detail-on-demand hides details in general and provides convenient means to get them back upon request. Apparently the drill-down process should allow iterative operation so each time the process performed, the data in focus is further narrowed down to a smaller subset of itself.

5.1.3 Multi-view Linking

Based on the theory of the visual working memory and the pre-attentive perception of human perception system, that human beings can rapidly perceive certain types of visual patterns in parallel and store a limited amount of them into the short-term visual working memory, we can take advantage of human brain by feeding it with multiple views at the same time, as long as the visual patterns in those views are carefully addressed by pre-attentive features and the visual working memory is not overloaded.

Though human brain can also establish relationships among multiple views, it requires cognitive efforts which dramatically drain the bandwidth used to access visual working memory. To avoid unnecessary consumption of the preciously limited perception resources, pre-attentive features are introduced again to link multiple views together in the pre-attentive processing, which is a lower-level than the cognitive processing in the reference model of visualization.

Moreover, when the information indeed overloads the visual working memory, which is very likely to happen due to the complexity of security incidents, there should be some ways to buffer the views temporarily so that the human users can concentrate on a few views and process them efficiently. This is the same as the RAM cache – Disk buffer solution in common computer system architecture. By this way the capacity deficiency of visual working memory is walked around.

5.1.4 Procedure Extraction

By using VizSEC system to explore network anomalies, the exploring procedures themselves are valuable knowledge already, because the security-domain-specific semantic interactions contain the reasonings that reveal the characteristics of the target anomalies. As pointed out by Lakkaraju et al. in [40], such exploring procedures play the same part as the sophisticated mathematical algorithms, in terms of conveying the knowledge discovered during the exploration of data between human and computer. In other words, common representation of exploring procedures both used in human cognitive model and computer analysis model close the problem solving loop on the cognitive level as described in Section 3.2.1. For example, if the result of an exploration is a pattern that data points form a line in a scatter plot of two attributes such as A and B of a dataset X, the exploration consisted of the procedure and the result can be described as “scatter plot dataset X by attributes B vs A → line pattern”. Apparently this description can be understood by human immediately. As for the computer, the procedure part can be easily converted to a script such as “iplots(XA, XB)”, and the result part can be represented by a screen shot of the resulting view. Some times the exploration can be extract to mathematical models, which can speed up the analyzing procedures and enhance the analyzing results.

In addition, by using such common representation of exploring procedures, human users can exchange knowledge and experience more conveniently in the same “language”. This is an indispensable requirement in the security management community.

5.2 Principles

First of all, UniVis is a prototype VizSEC system mainly for concept demonstration. With very limited time, fast development is of the utmost priority. Upon this base, the system should be kept as simple as possible, while other criteria such as functionality, scalability and usability can be traded off as long as they are sufficient for our experimentation.

In order to meet the requirements, we refer to the pros and cons of existing VizSEC systems for the best practices. In addition, we attempt to separate each component of the system by modular design. As the components are loosely coupled to each other and communicate via common interfaces and protocols, each component can be improved without affecting too much on the

whole system.

Lastly, popular open source projects should be utilized if possible, whereby they can be better adopted by direct modification in the source codes, and the popularity providing rich sources to solve the practical problems during the implementation.

5.3 System Architecture

Figure 16 shows the system architecture according to the design framework of the VizSEC system with slight adjustment. Such architecture mainly tells the data flow in a complete system working process converting dataset to visualizations, and the position of each component in this data flow. In general, the ellipses indicate input and intermediate data files, the boxes with round corners represent output files consisted of plot snapshots and extracted parameters used to generate the plots. Meanwhile square boxes except for the RAM represent system components either processing the data or coordinating the work of other components.

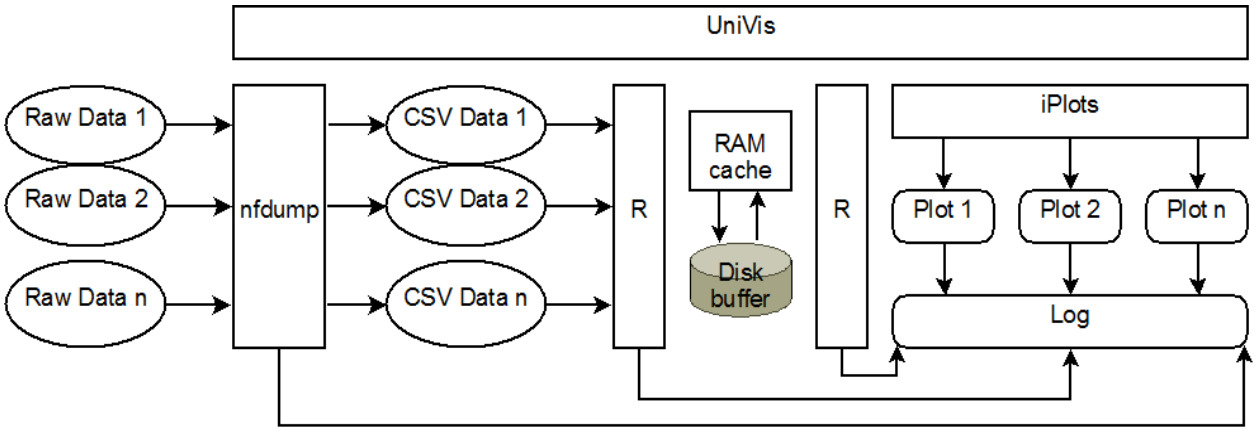


Figure 16: UniVis system architecture

A complete visualization process by using UniVis starts from the multiple input data files on the most left side of the frame. These raw binary data files are provided by nfcap, which is part of the open sourced toolset named NfDump[41] and plays the role of a NetFlow collector.

The main utility of NfDump is nfdump with command line interface (CLI), which is responsible for dumping the raw binary data files to the comma separated value (CSV) data files. Preprocessings such as aggregation, anonymization, and prefiltering are also performed by nfdump during the dumping process by additional dumping options. Implemented in C programming language, nfdump is optimized in processing speed. Hence it is selected for both the preprocess and the prefilter modules in the framework.

The R[42] is a very popular open source statistical analysis environment featured by plenty of statistic and graphic libraries and full data management support. R provides a complete programming language in its computing environment and works efficiently as it is implemented in the C programming language. As depicted in Figure 16, R is used in two places of the system.

The left R component is an additional component comparing to the original framework, due to

the additional intermediate CSV data between the nfdump component and the R component. This R component simply works as a data reader to import the CSV data files to the R environment. On the other hand, the right R component is employed mainly for data manipulation and statistical computation. The postfiltering and the encoding modules is fulfilled by its data manipulation functions. In addition, it also apply statistical processing to the data. As the data is already loaded into the RAM, an new module possibly called postprocessing can be added to the framework between the postfiltering module and encoding module, in order to provide interactive computation.

Obviously, the graphic engine module is realized by the iPlots[43] component. The iPlots is a Java interactive visualization toolkit that is incorporated with the R environment. In order to be used with R, iPlots includes a set of R scripts that provide interfaces for the R users to utilize the interactive plotting library transparently without knowing any Java API. On the other side, the iPlots can take the data stored in the R environment as input for the visualization.

Each plot view generated by the iPlots is a Java window frame featured with several interactive operations. Such views can be logged as snapshot image files, together with text files recording the parameters used in the previous components. Organized in a hierarchical tree structure with specific naming scheme, those log files keep the tracks of the exploring procedures.

Finally, above all the above components, the frontend component is filled as the user interface of the whole framework. By extending the length of the box representing this user interface module, we allow the frontend to interact with preprocessing and prefiltering modules as well by design. In this way all essential components can be operated via an integrated GUI control panel named UniVis.

5.4 Component Composition

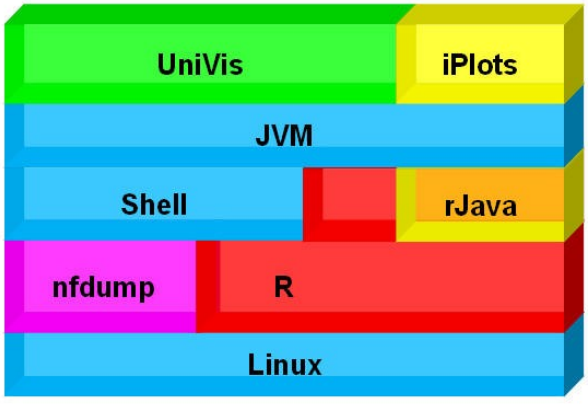


Figure 17: UniVis component composition

Figure 17 gives an overview of how each building block in the UniVis depends and interacts with each other.

The blue blocks represent the common environments via which other components communicate. The Linux block is the operating system (OS), providing basic OS functionalities such as file system access.

Right on top of the OS lie two core components: nfdump and R, since they are installed directly on the OS. Both of them can be launched directly from the Shell environment.

The nfdump is a common CLI utility that runs each time a valid command is sent to the Shell. The control panel interacts with nfdump by executing nfdump commands in the Shell via the Java Runtime in the Java Virtual Machine (JVM).

The R is a more complex environment that has its own command line interface and language. The control panel interacts with the R in two ways. The first way is to use R in BATCH mode as a common CLI utility. In other words, the control panel uses the R in the same way as it uses the nfdump. The second way is to use the R via rJava, which provides interface between the R environment and Java programmes running in the JVM, and is based on the Java Native Interface (JNI) framework. In the second way the control panel launches the R environment by creating an instance of R engine internally, and maintains the R environment by implementing an interface that handles the R main loop callbacks. In brief, the control request its R engine to evaluate R scripts, and the evaluation result is returned by the callbacks. Data type conversion is required during the questioning-answering mechanism. This is partially addressed by rJava.

Finally on the very top layer lie the Java components including the GUI control panel and the graphic engine iPlots. Though they are both Java components and iPlots can be indeed used directly by the control panel, the iPlots here works more closely with R via rJava because its input data for visualization comes directly from within the R environment. Therefore, the iPlots and the control panel work independently and communicate with each other via the R environment.

5.5 Functional Module Description

Previously we discussed the system from the perspective of its compositional entities. In this section the system is described in more details in terms of its functional decomposition.

5.5.1 Data Source Manipulation Backend

This module takes care of the import processing of the NetFlow data. In all, the import processing consists of two conversions between three data types. A brief comparison of the data types is given in Table 5.1. No database is used for the storage and manipulation of data but only a collection of data files and some processing utilities, because the latter solution is simpler to implement for a prototype and fast enough for the experimentation.

File Type	Producer	Format	Compression Algorithm
NfDump Raw	NfDump	Binary	Fast LZO1X-1
CSV	NfDump	ASCII Text	None
R Workspace Image	R	Binary	gzip DEFLATE

Table 5.1: Data types processed by the data source manipulation backend

As a starting point of the whole visualization processing of the UniVis, the NfDump raw data is normally supplied by the nfcapd utility of the NfDump toolset installed on the NetFlow collector. In addition, the raw data can also be produced by other utilities of the NfDump toolset such as nfdump, usually with additional processing applied at the same time such as filtering and aggregation. The NfDump raw data is stored in binary format using fast LZO1X-1 compression, which does not offer high compression ratio but extremely fast compression speed.

Taking the NfDump raw data as input, the first data conversion outputs the CSV data in ASCII text format. This is the only human readable data type during the whole data manipulation phase, and presents the table structure of the NetFlow data in an intuitive way, in which each row represents a data record and each section of a row separated by commas represents a data attriubte column. We have defined table data in Section 3.2.1 and stated in the same place that NetFlow data is a kind of table data. As shown in Figure 18, the first row of every CSV file shows the abbreviations of the names of the NetFlow data attributes, which are selected according to the discussion on the network traffic measures employed by the NetFlow technology. These rows are called the title rows in this report, and the rest of the rows are called the data rows.

ts	tss	te	tes	td	pr	prs	sa	sas	sp	da	das	dp	flg	flgs	pkt	byt	bpp
1207922656.149	2008-04-11 17:04:16.149	120	200	393.472	6	TCP	2653178181	158.036.073.069	2004	141	084	3306	0	.....	13	691	53
1207922656.207	2008-04-11 17:04:16.207	120	200	369.728	6	TCP	1410394788	084.016.234.164	3306	265	158	2004	0	.....	15	1515	101
1207925501.909	2008-04-11 17:51:41.909	120	200	33.152	6	TCP	2653178181	158.036.073.069	1026	113	067	3306	0	.....	9	462	51
1207925502.101	2008-04-11 17:51:42.101	120	200	33.088	6	TCP	1131257971	067.109.160.115	3306	265	158	1026	0	.....	10	746	74
1207925539.213	2008-04-11 17:52:19.213	120	200	9.024	6	TCP	2653178181	158.036.073.069	1060	364	217	3306	0	.....	3	144	48
1207925564.815	2008-04-11 17:52:44.815	120	200	8.896	6	TCP	2653178181	158.036.073.069	1070	326	194	3306	0	.....	3	144	48
1207925589.773	2008-04-11 17:53:09.773	120	200	8.960	6	TCP	2653178181	158.036.073.069	1081	326	194	3306	0	.....	3	144	48
1207925614.693	2008-04-11 17:53:34.693	120	200	1896.448	6	TCP	1410394788	084.016.234.164	3306	265	158	1125	0	.....	49	3236	66
1207925614.693	2008-04-11 17:53:34.693	120	200	1896.448	6	TCP	2653178181	158.036.073.069	1125	141	084	3306	0	.....	29	1613	55

Figure 18: Example CSV data table

It is worth mentioning that nfdump dumped two kinds of text data of the same NetFlow data attribute. One in the integer format is for machine processing, and the other one in string format is for human processing. For example, the Ipv4 address string “158.36.73.69” can be represented by integer “2653178181”. Evidently attributes that are essentially numeric have the same value in both formats. In our case we dump both formats for the non-numeric attributes, and dump only integer formats for the numeric attributes.

We choose such solution instead of only using integer format and converting them to string format dynamically on demand, because the string data and the integer data can result in two visualization optimization, namely, continuity and trustworthiness. Continuity means that the data points should keep near to each other in the visualization if they are near originally.

Trustworthiness means that data points should be near to each other originally if they are near in the visualization. Continuity and trustworthiness are two optimal goals often applied in multidimensional reduction algorithms.

String data is reckoned as ordinal data, whose order is determined by the position of their characters in the ASCII code. For example, 'a' is ahead of 'b' because 'a'(97) is ahead of 'b'(98) in their ASCII codes. Also, characters that are ahead in the string have higher priority in determining the order of the strings. For instance, “ab” is ahead of “bb” since the first character of “ab” is ahead of the one in the same position of “bb”. The distance between each two data values only represents the number of different observed data values between these two values. An easy observation of ordinal data is that no gap will appear between two neighbor values even though there are potential values that once observed, can be inserted between them according to the ordering rules. For example, only two strings are observed, such as “a” and “c”. Unless “b” is observed, “a” and “c” is still neighbor strings. Therefore, ordinal data have good continuity but probably bad trustworthiness, such as the string data.

On the other hand, integer (or numeric) data instinctively keep the distance between each data values. For example, the distance between '1' and '3' is always 2 as long as they are reckoned as integer number. Thus, integer data have good trustworthiness but probably bad continuity. By keeping both of them, we may see the difference between two optimal visualization of the values of the same attribute, hence acquire a more complete understanding of the data. Obviously, more memory space is required.

The conversion from the NetFlow raw data to the specific CSV data is carried out by a slightly modified nfdump utility, because the original nfdump can not fulfill our needs for both integer and string output at the same time.

As for the conversion from the CSV data to the R workspace data, the system can work anyway without it because R can directly read CSV data each time. The main reason to add this conversion is to improve the processing speed and to save storage space.

In the R environment data and functions are seen as objects in its workspace. The R can save the objects in such workspace onto disk as a workspace image file, and load them back from the image file to the workspace. Compared to the CSV data file, the workspace image file can be loaded much faster into the R environment, and consumes much less disk space because of the utilization of the compression and the binary format. Thus, though converting the CSV data to the R work space image is an additional process, it is worthwhile as long as the data is going to be used more than once, which is true in practice.

5.5.2 Visualization Preparing Components

This module covers all kinds of preparing processes of the input data before the visualization process. In general, this module produces data of new attributes from the data of existing attributes of the netflow data, which can be both visualized later by the interactive plotting components described in the next section. In other words, the new data produced by this component is multivariate data because the new attributes it belongs to depend on the existing attributes.

Many kinds of method can be used to produce such data of certain new properties, such as multidimensional reduction, cluster analysis, spectrum analysis, correlation analysis, and so on.

As described in Section 3.2.4, multidimensional reduction method such as the principle component analysis (PCA) helps to prepare the original high dimensional data in a much lower dimensional subspace without losing much original information, in order to allow lower dimensional plotting schemes to directly visualize the data. The algorithms of clustering analysis such as multidimensional scaling (MDS) are similar to multidimensional reduction methods to some extent, but focus more on preserving the proximity of data points rather than optimize the variances. In this way the cluster characteristics of data hidden in a high dimensional space can hopefully be preserved and observed when projected to a lower dimensional subspace.

Different from multidimensional reduction and cluster analysis that attempt to discover the spatial characteristics, spectrum and correlation analysis concentrate on the temporal characteristics. Spectrum analysis methods such as fast Fourier transform (FFT) transform the data of a single attribute observed along time to the spectrum domain, so the spectrum energy distribution can be shown by a scatter plot, in which the x axis represents the frequency and the y axis represents the energy. Correlation analysis targets to find the similarity between two sets of observations by feeding them to the correlation function and evaluate the output.

Apart from the above methods, more methods can be added as long as they generate new attributes from the original ones. Because R provides rich data manipulation and statistical computation functionalities, each methods can be implemented separately as functions in R scripts. Such plugin style structure results in great extensibility of this component, by which users familiar with the R environment can easily implement customized functions by themselves.

5.5.3 Interactive Plotting Component

This module is responsible for all plotting and interaction functions of the data ready for visualization. Because we use the R to manage and prepare the data for visualization, the most efficient way to use the data is to use them within the R environment directly. As stated in Section 5.4, the iPlots (interactive plots) is the very utility we look for.

Exactly as it is named, the main functionality of the iPlots is generating interactive plots. It is implemented in Java because of the advantages such as platform independence, object orientation and graphic capabilities. Besides common interaction functions such as zooming and dynamic querying, the iPlots distinguish itself from other interactive plotting toolkits by its linked plotting ability.

In order to provide the linked interaction features, the iPlots brings in the concept of iSet. Briefly, an iSet is a data structure consisting of arbitrary number of variables, which are vectors of the same length. In other words, the iSet perfectly represents the MDMV data by using its variable vectors to represent the data attributes. As shown in Figure 19, the iSet serves as root objects of the plots in iPlots. From each iSet, arbitrary plots can be generated, but each plot can only be linked to one iSet. Under such hierarchical structure, common parameters and temporary data of plots linked to the same iSet can be attached to the iSet and shared by all the plots. For example, each iSet maintains a color index for every data points it contains. Every plot linked to

the same iSet then uses the same color index to brush the data points displayed in its own view. This is indeed how linked color brushing is realized. Also, each iSet maintains a selection index to save the identification of each selected data points. Every plot linked to the same iSet then uses the same selection index to identify and highlight the selected data displayed in its own view. This is how linked selection and highlighting is achieved. In the same way, new attributes generated by the visualization preparing components are also be attached to iSets dynamically so they can be shared and displayed in a linked manner by multiple plots as well.

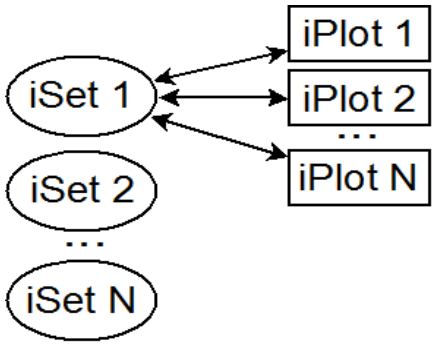


Figure 19: iPlots internal structure

In addition to the interactive plotting, the iPlots can be used inside the R environment. Users can call the iPlots functions by directly using the R language regardless of the Java implementation details of the iPlots. As mentioned before, such transparent operations from R is achieved by using the rJava package, which provide interaction interface between R and Java via the JNI framework. Thanks to such transparency, we only need to write R scripts in order to create and manage the data set and their plots in the iPlots. Further interactions are simply performed directly over the Java windows showing the plots as well, such as adjusting the parameter for the alpha channel, which controls the transparent level of the objects in the visualization. Once any new plotting schemes are supported by the iPlots, they can be easily added to the UniVis simply by adding new R scripts. Thus this module is also highly extensible.

5.5.4 GUI Control Panel Frontend

This module is the GUI frontend of the UniVis system. It connects all other modules in an integrated control panel, whose design involves the general visual exploration procedure of the netflow data.

Coming after the control panels of popular image processing softwares such as GIMP, this control panel is designed for the best operating efficiency in a way that all operations are exposed on one scroll panel without any hierarchical interaction techniques such as menu or dialog box. In order to guide the general visual exploration procedure, operations of each system functional module are grouped together, and all modules are arranged sequentially according to the processing flow. Figure 20 shows the basic layout of such control panel.

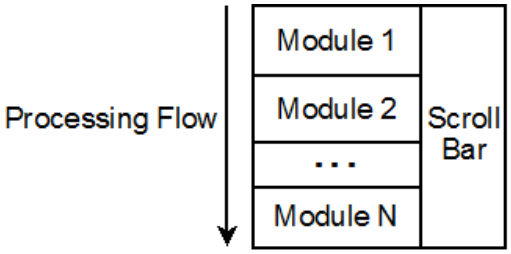


Figure 20: GUI control panel layout

Each control block of the corresponding module in general should provide intuitive interface to handle three types of operation consisting of the input selection, the parameter setup, and the final execution.

For the control block of the data source manipulation module, it should provide convenient interface to utilize the functions provided by the nfdump backend. For example, in the selection of the input raw netflow file, a static file explorer can be used to locate the files more efficiently than the pop-up explorer. For the time window selection, an initial time window retrieved from the selected netflow files can be a good reference to specify the customized time window. As for the filter expression, a feature to validate the correctness of the expression can help to reduce operational mistakes. These features are generally not available in the original CLI backend. Finally, there should be interfaces for the execution of the data type conversions.

For the control block of visualization preparing module, it should provide interface for the netflow attribute selection, the preparing method selection, the preparing parameter setup, and the execution of the preparing process.

The control block of the interactive plotting module is the main part of the control panel. In addition to the common interfaces for the netflow attribute selection, the plotting method selection, the plotting parameter setup and the plotting execution, this block introduces the concept of the visualization tree in order to visually represent the relationships between the visualized datasets and their plots, and the relationships between each visualized dataset.

The idea of the visualization tree is initially inspired by the hierarchical internal structure of iPlots as shown in Figure 19, where an iSet managing the data to be visualized can be seen as the tree root, whereas an iPlot managing the view generated from the root data can be seen as the tree leaf. Moreover, because of the basic requirement of VisSEC system on details drill-down, the dataset under analysis each time usually gets more focus on some local interesting spots, hence temporary subsets of the original dataset need to be generated accordingly. As the original data is the root, its subsets can be seen as the branches.

Apart from depicting relationships between the datasets and the plots, and between the dataset and its sub-datasets, the visualization tree can be utilized for recording the exploration procedures. Each plotted view, each generation of the sub-dataset, and each color encoding of the data are parts of the exploration results. By logging these essential objects and actions, not only the exploration results can be stored, but also the exploration procedures can be traced and added to the knowledge base which can be used in the future exploration. In other words, the

visualization tree transforms visual exploration procedures to symbolic rules, which can be re-applied to new datasets. By bridging the human cognitive model and computer analysis model, the visualization tree closes the human-computer problem solving loop between pattern discovery and pattern searching, as stated in [40].

Finally, because many processing take quite some time and many operations of the components can be carried out independently, parallel processing mechanism can be employed in order to provide a more efficient control panel. For example, when some new raw netflow data is being dumped, the whole system should not be blocked until the dumping process is finished, but allow the users to operate on the previously dumped data. Additionally, because the raw netflow data usually arrives at a certain interval in practical using scenario, scheduled routine operations can reduce a lot of man hours. A schedule table showing the scheduled operations can be a good feature. As most of the operations are launched by the scripts, this feature is be feasible for the implementation.

6 Implementation

According to the design in the previous chapter, this chapter describes the implementation details of the UniVis according to its functional modules. Discussions on the deployment, usage and performance issues are followed behind.

6.1 Data Source Manipulation Backend

The implementation of the functional module of the data source manipulation backend includes three parts: 1) the modification of the source code of nfdump for dumping raw netflow data to CSV data, 2) the R script for the data conversion from the CSV data to the R workspace image data in the R batch mode, and 3) the shell scripts for the execution of the manipulation processes.

For the first part, we achieve our goal by hacking into the source code of the nfdump to add one more predefined output mode for its “-o” option, and add the necessary codes for the new output mode referring to the existing codes for other modes.

For the second part, we use two R functions for the conversion. Function “read.table()” reads the CSV files to the R internal data object, while function “save.image()” converts and saves all objects in the R workspace to the R workspace image file.

Finally, shell scripts “dump.sh” and “preload.sh” utilize the above functions to carry out the dump and preload processes. Notice that script “reload.sh” uses the R in batch mode in which the R is used as a simple processing utility than a full functioning environment. In this mode, the R takes in the input data and the parameters, and produce the output once per command in the Shell. As both scripts run independently in the Shell, they can be performed in parallel without interfering each other.

6.2 Visualization Preparing Components

First of all, this module implements the data object management functionality within the R workspace. By directly utilizing several basic R functions, the implementation of such functionality is a matter of writing the R scripts.

The implementation of the visualization preparing module is the scripts of R function as mentioned in Section 5.5.2. Because the lack of time and the initial attempts to apply some preparing methods such as PCA do not end with good result, the implementation of this module is left for future improvement.

6.3 Interactive Plotting Components

As mentioned in Section 5.5.3, the actual functionality of interactive visualization is implemented by the iPlots toolkit, whereas our implementation is writing R scripts to control the plotting created by the iPlots.

More scripts for other plotting and other functions such as subset creation, color encoding and so on are attached in the appendix. Notice that these scripts are in fact hard coded in our implementation due to time limitation, though they can be easily extracted as R script files, which are more modular but need to be loaded and unloaded to the R environment each time being used. This is because the batch mode of R can not be used here due to the complex context which each R command is executed in and which can only handled in full functioning R environment.

6.4 GUI Control Panel Frontend

The figures from Figure 21 to Figure 24 present all function blocks within the GUI control panel implemented using the Java Swing GUI framework

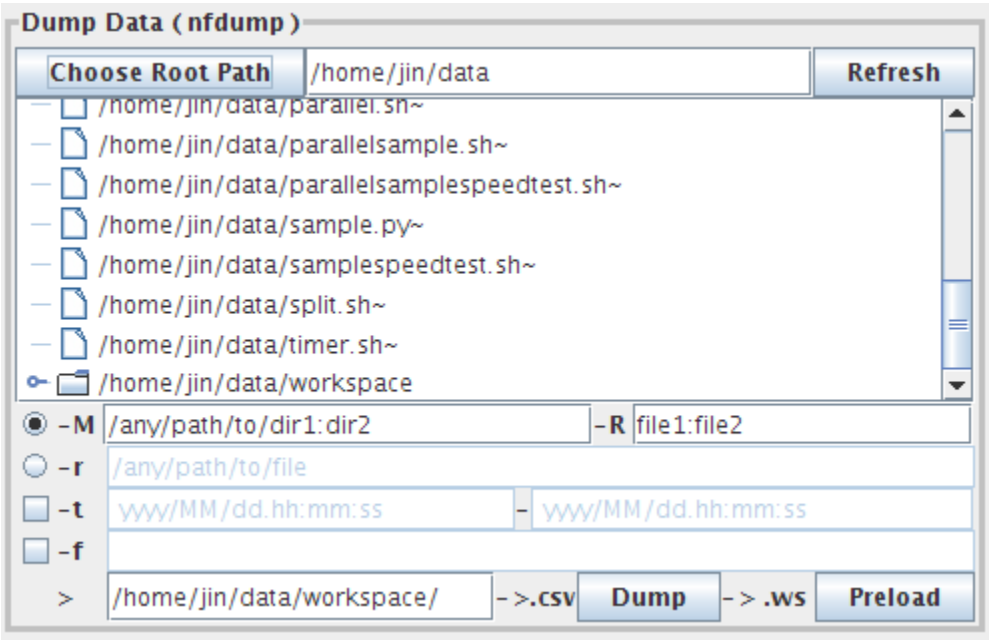


Figure 21: GUI control panel block A

Block A as shown in Figure 21 corresponds to data source manipulation module and mainly offers convenient interface for setting parameters for nfdump.

The “Chose Root Path” button and the text field on the right of it are used to decide the current working directory for the nfdump. The file tree right below them shows the file system structure under the specified directory, so users can navigate and select the raw netflow data files they want to use in the nfdump.

The labels start with a dash indicate the dumping options of nfdump. The first two rows starting with a radio button determine the path of the selected input raw netflow data file(s). Because they start with radio button, only one row can be enabled at a time. The first row can specify multiple files, while the second row can only specify a single file. Multiple files are set by defining the starting file and the finishing file, whereby the file sequence are recursively sorted according to the directory names or file names. Such method of selecting multiple files is useful because in

practice the raw netflow data files are usually organized in a directory structure such as “Router ID\Year\Month\Day\filename”, and the files are named in the default format as “YYYYMMDDhhmmss”.

The “-t” row is used to specify the time window, so only the flows within such period are processed. If the choice box before the “-t” label is ticked when a file or a range of files are already selected, the text fields followed will show the time window of the selected file(s).

The “-f” row is used to specify the filter expression for the filtering of the flows. For example, “host src 192.168.0.1” tells the nfdump to only process the flows whose source IP address equals to “192.168.0.1”. The filter syntax is very similar to the one used in tcpdump. More detailed instructions on the usage of the filter and other opitons can be found in the manual document contained in the nfdump package.

After the parameters for the options are set, the last row specify the output file name except for the suffix, which is automated added to the output files generated by clicking on the “dump” or the “preload” button. Intuitively, the“dump” button dumps the raw netflow files to the CSV files, while the “preload” button converts the CSV files to the R workspace image files.

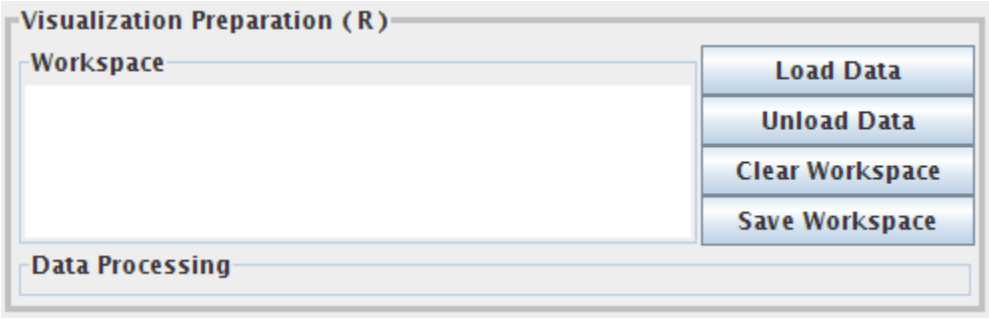


Figure 22: GUI control panel block B

Block B deals with the preparing processes before visualization. As shown in Figure 22 there are two sub-blocks. The first one manages the data objects in the R workspace, such as loading new dataset, unloading a single data object or clear all data objects, and saving all data objects into the workspace image file. Loaded data objects in the workspace will appear in the list on the left of the buttons.

The second sub-blocks is left empty because of the reasons given in Section 6.2.

Block C is obviously the most complex control block on the control panel. All visualization operations are gathered here. As shown in Figure 23, the whole block is also divided into several sub-blocks.

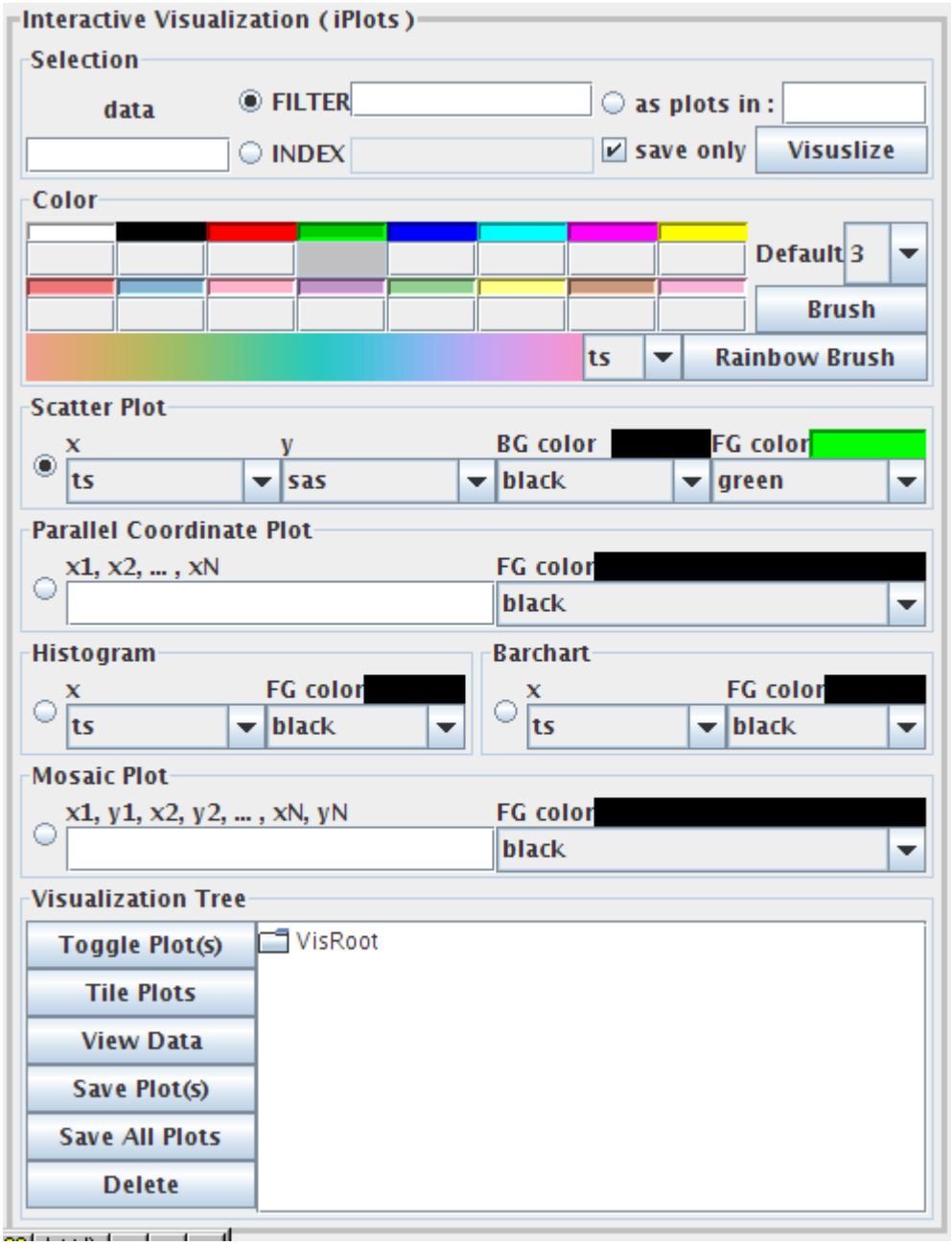


Figure 23: GUI control panel block C

The “Selection” sub-block is mainly used to specify the dataset to be visualized. The text field under the “data” label shows the dataset contained by the current iSet. As stated in Section 5.5.4, the dataset can be the original dataset directly converted from the CSV data. This kind of dataset is called the root data. The dataset can also be the subset of other dataset. This kind of dataset is called the branch dataset. There are two ways to generate the branch data, which are labeled as “FILTER” and “INDEX”. The “FILTER” way uses the filter expression to take certain subset

from the current dataset. Such filter expression follows the R syntax, but is also very similar to the nfdump or tcpdump filter syntax. On the other hand the “INDEX” way allows the users to pick up data points directly from the plotted views by the mouse. In short, the “FILTER” way is good at precise but inflexible subset selection, whereas the “INDEX” way is good at flexible but coarse subset selection. The rest of the three GUI components in this sub-block are used in the end of the overall visualization procedure, so they will be described later after those “plotting” sub-blocks.

The “color” sub-block is responsible for the color brushing. Two color brushing methods are supported currently. The first one is the filter based color brushing, which is on the upper half of this sub-block. The main GUI components of this brushing is the 8x2 color palette. Under each color label is a text field, where the filter expression can be filled in, so the data records meet the filter condition will be brushed with the color accordingly. The “default” combo box determines the basic color for the data records not meeting any of the filter expression. When the “brush” button is clicked, the system sequentially brush the data with certain color according to the filter assigned to the color. The brushing sequence is defined by the color positions in the palette, where the first color starts from the left-top and the last color ends in the right-bottom. In other words, if the datasets filtered out by two filter expressions are overlapped, the filter assigned to the color with a latter position in the palette brushes the overlapped data with its color.

The second color brushing method is the “rainbow brushing”. This brushing based on a continuous color space, in which the color gradually changes from one color to another. There are many ways to use the rainbow color space, but in our case, we first select a netflow attribute from the combo box on the right of the rainbow label. Then, the total amount of different values of this attribute is counted and used to evenly split the rainbow color space. Thus, each section of the split color space can be used to brush a value of the attribute. In our implementation, because we use 64 discrete samples of rainbow color space instead of a real continuous rainbow color space, when the number of different values exceeds 64, we can not provide different color for each different values any more. Instead, we evenly split the number of different values to 64 parts, and brush the values in the same parts with the same color. Notice that in our implementation it is always the number of different values is considered instead of the actual number space. In other words, we treat numeric numbers as strings. Finally, when the “rainbow brush” button is clicked, the dataset shown in the text field below the “data” label within the “selection” sub-block will be brushed with the rainbow colors according to the value of the selected attribute. As long as the dataset is loaded, the color brushing can be carried out regardless of the existence of plots.

Then we have the plotting sub-blocks, which provide interface for the plotting parameter setup of each plotting function. Intuitively, the labels of “x”, “y”, or “xN”, “yN” represent the axes in each plotting scheme, which has been introduced in Section 3.2.2, except for the Mosaic plot. This plot is only for testing and does not appear scalable. As it will not be used in our experimentation in Chapter 7, we do not introduce it in this report either. The BG and the FG color means background and foreground colors used in the plot. When the specific color brushing provided in the “color” sub-block is applied, the colors specified here will be ignored. At any time only one plotting scheme can be selected by selecting its radio button.

Finally, when the data, color brushing, and plotting scheme are all set, the system is ready for the visualization. When the “Visualize” button on the “selection” sub-block is clicked, the control

panel tells the iPlots library to plot the specified view via the R. When the plotting is completed, it become a leaf node of its data set. Such internal structure will be visually presented in the “Visualization Tree” sub-block, where the data set and plots are displayed and managed in a tree structure.

Those buttons on the left side of the “Visualization Tree” sub-block provide some practical functions for the user to manage the visualization tree. After selecting a node in the tree by the left click of the mouse, clicking the “Toggle Plot(s)” button hides or shows the plot node or the direct leaf plot nodes under a data node; clicking the “Tile Plots” button spreads all the plot nodes under a data node with equal area on the screen; clicking the “View Data” button should pop out a window showing the text details of the data node in a traditional table view, however it is not implemented in our current version yet; clicking the “Save Plot(s)” button saves the image of the plot node, or save the filter expression and color brushing expression of the data node together with the images of the plot nodes directly linked to it; similarly, clicking the “Save All Plots” button saves all data node information and plot node images within the subtree under the selected data node; finally, clicking the “Delete” button removes the selected node and the whole subtree under this node.

Apparently, the “Save Plot(s)” and the “Save All Plots” buttons actually realize the logging function. Because when the information of each data node and the image of each plot node is recorded, the exploration procedure is logged as well. In other words, previous procedures can be stored as knowledge and reused in the future exploration. Because manually reproduction of a previous procedure is tedious, an additional feature of the visualization tree allows the user to copy the plotting procedure of any data node in the visualization tree. All that the users need to do is to select the radio button of “as plots in” in the “selection” sub-block, right click on the original data node in the visualization tree, and click the “Visualize” button in the “selection” sub-block.

The last feature for block C is that by ticking on the “save only” choice box in the “selection” sub-block, all new plots will be minimized and hidden. This may save a considerable time if the user only need to log the exploration procedures and the results rather than interactively carried out the exploration.

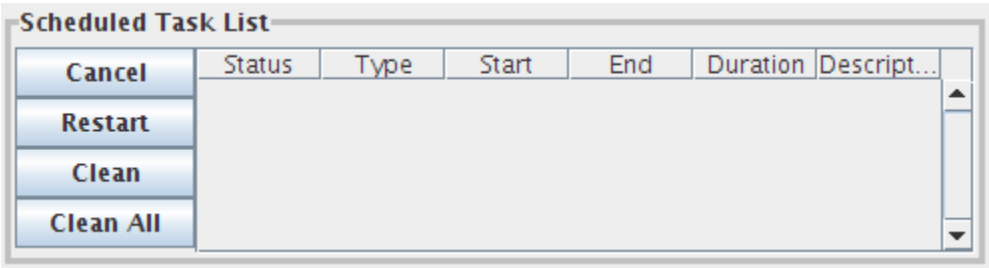


Figure 24: GUI control panel block D

As mentioned in Section 5.5.4, many processes in this system can be carried out independently, hence parallel processing mechanism may help to improve the efficiency. The last control block D provides the interface to monitor and manage such mechanism.

Internally, the UniVis system maintains two task managers. One for the Data manipulation tasks,

which can be executed concurrently. The other one for the rest of the operations, which can not be performed in parallel. Though these two task managers run independently, the tasks within the same manager still run in sequence. So, the table on the right side of the block D dynamically displays the status of all the tasks managed by either manager. In addition, the buttons on the left side of block D provide several management functions for the tasks displayed in the scheduled task list. For example, as long as a task is not in the “INPROCESSING” state, the “Cancel” button can call off the upcoming execution of the task. On the other side, the “Restart” button can always reschedule the canceled task to be executed later, because the task manager keeps checking the status of its tasks in order to decide if any task needed to be executed. Lastly the “Clean” and the “Clean All” buttons are for the convenience of removing the failed or completed tasks in the list in order to get a clearer view.

6.5 Deployment

In Figure 6, the UniVis is deployed on the NetFlow Analyzer, which can be any computer terminal with sufficient computing and graphic capability. The deployment of the UniVis requires some efforts, but should not be difficult.

First of all, a latest Ubuntu Linux operating system (OS) should be installed. This is because we use the Ubuntu Linux OS as our implementation environment. In fact, the UniVis system can be deployed on any Linux OS based computer system. However, we strongly recommend to use the latest Ubuntu Linux OS to get the latest upgrades of the supporting utilities of the UniVis, in order to get the most features and the least troubles. The latest Sun Java Software Development Kit (SDK) over should be installed also.

Second, the NfDump toolset needs to be installed. Because the UniVis needs specific CSV data with customized netflow properties, which can not be dumped by the original nfdump, we can not use the NfDump package offered by the software package repository of the OS directly. Instead, we need to download the latest NfDump source code, modify it accordingly, and then compile and install it.

Third, the R environment have to be set up. This can be done directly by installing the necessary R packages appeared in the software package repository of the OS. It is recommended to add the Comprehensive R Archive Network (CRAN)[44] as an additional source of the software package repository specifically for the R, in order to get the latest R packages.

Forth, after the basic R environment is prepared, it is time to install the iPlots. Because the iPlots is designed to be used with R, and the R environment offers a very convenient software management framework similar as the one of the OS, the iPlots can be easily installed directly via some R commands. The simplest way is to install the Java GUI for R (JGR)[45] package so all necessary R packages including the iPlots and other packages it depends on can be installed automatically.

Finally, we simply copy our implementation of the GUI control panel of the UniVis to the target computer. As this control panel integrates the functions of the above softwares in one place, we can launch the UniVis system by executing the “univis.run” script included in the package from a terminal.

6.6 Example Use Case

With a complete UniVis system successfully set up, this section further describe a common use case of the UniVis by a simple exploration of the NetFlow data.

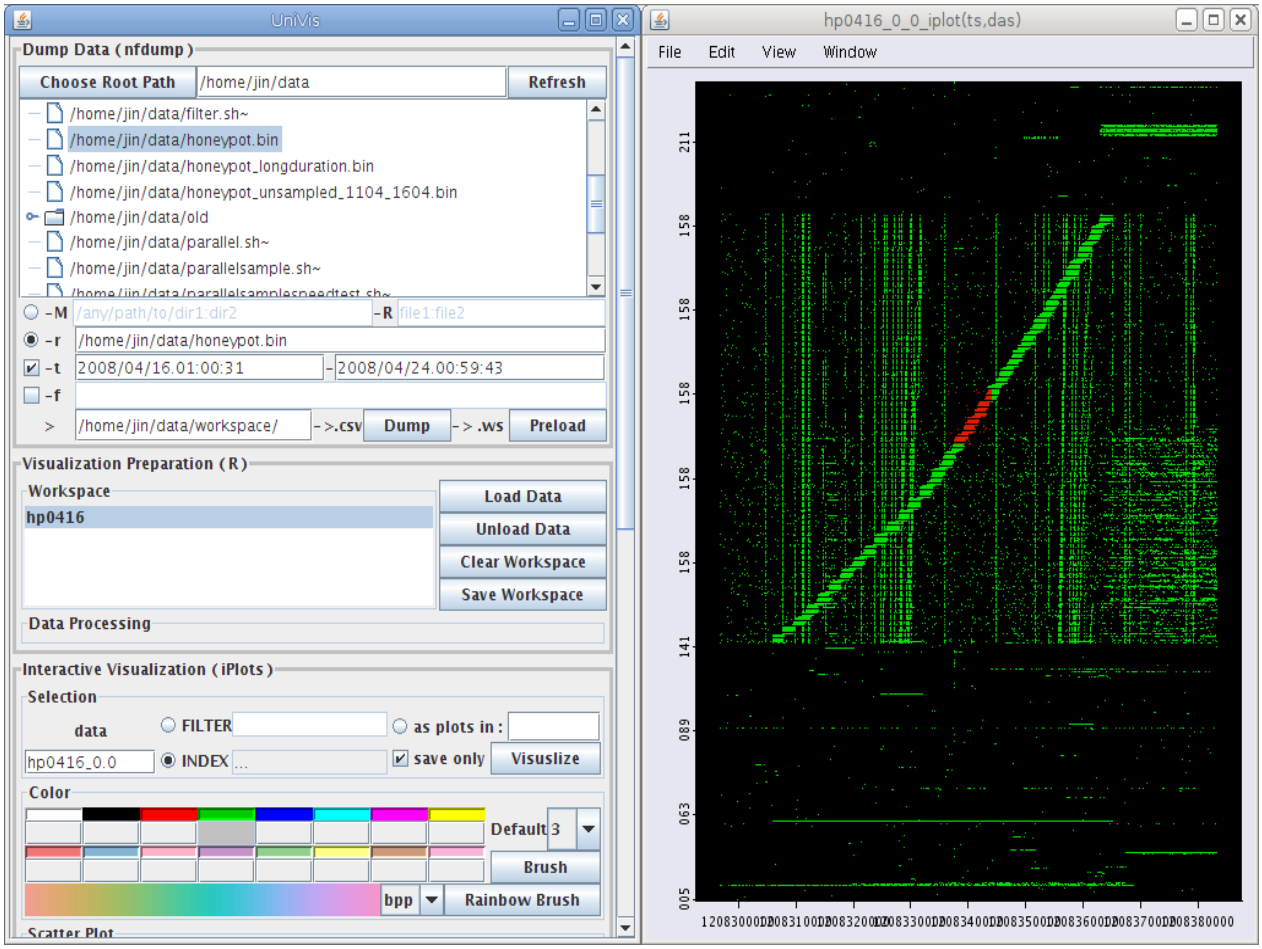


Figure 25: Using scenario overview

According to the requirements discussed in Section 5.1.1, the UniVis should provide a global overview of the input data. Figure 25 shows that the UniVis gives such an overview on the right of the control panel. In fact, generally all visualizations of input data are going to be presented on the right of the control panel, whereas the control panel itself keeps on the left. In Section 5.5.4, we design the control panel in a way that the control blocks of necessary components are organized from the top to the bottom according to the basic exploration procedure. Hence we start from the top of the control panel to introduce such procedure.

First of all in the “Dump Data” block, we preprocess the raw data and convert it to the R workspace image. We start from selecting the radio button of “-r” and then choose a single raw netflow data file named “honeypot.bin” from the file tree. When the mouse is over the raw data file in the tree, the amount of flow records within this file is shown in a pop-up tooltip. Usually

the amount under 500 000 is considered best for the performance of the system, depending on the processing power of the computer. Because the flow number of the file is under the limit, we can use all the data without considerable performance degradation. Otherwise we have to narrow down the flow number by applying the time window or the filter. By selecting the selection box of “-t”, we can see the time window of this raw netflow data file is from “2008/04/16.01:00:31” to “2008/04/24.00:59:43”. Since we simply want to use all the netflow data contained by this file, we can leave the time window as it is or uncheck the selection box of “-t”. For the same reason we left the selection box of “-f” unchecked so we do not apply any filter to the data. After setting up the last parameter setup by filling in the output path in the text field at the bottom of this block, we can sequentially press the “Dump” button and the “Preload” button to carried out the actual processing and get the CSV data file and the R workspace image file. Simultaneously corresponding tasks named “DUMP” and “PRELOAD” are added automatically to the “Scheduled Task List” shown in Figure 26.

Then we move on to the “Visualization Preparation” block. Because no further data processing is implemented yet, the only thing to do here is to load the previously generated R workspace image file. This is done by pressing the “Load Data” button and selecting the R workspace image file from a pop-up file selecting window. At the same time a “LOAD” task is added to the “Scheduled Task List”. When the task is completed, the name “hp0416”of the loaded data appears in the “workspace” list, as we can see in Figure 25.

If we select the dataset “hp0416” in the “workspace” list, the name should appear in the text field under the “data” label of the “Selection” sub-block inside the “Interactive Visualization” block, indicating data set “hp0416” is going to be visualized. Then we use the “Scatter Plot” sub-block to set the UniVis to plot the dataset in the scheme of scatter plot, which uses the netflow properties start time (“ts”) and destination IP address string (“das”) as the x axis and the y axis respectively. Lastly, we click on the “Visualize” button in the same sub-block in order to generate such scatter plot. This action not only adds a “VISUALIZE” task to the “Scheduled Task List”, but also adds relevant nodes to the “Visualization Tree”. In this case, under the symbolic “VisRoot” node, we have data node “hp0416” representing the original data set, data node “hp0416_0” representing the first sub-dataset contained in an iSet and directly used for the visualization, and the plot node “hp0416_0_0” representing the first plot linked to the iSet “hp0416_0”. When the plotting process is completed, it is a minimized hidden window. In order to show it in a convenient size and position, we select the “hp0416_0” node and click on the “Tile Plots” button. As a result, we get the tiled “hp0416_0_0” plot filling up the space on the right of the control panel as shown in Figure 25.

Notice the strange thick diagonal line across the scatter plot “hp0416_0_0”, it indicates that a subset of IP addresses have incoming flows one after another along time. The thickness of the line means for each flow the duration is quite similar. This looks a “creepy-crawly” scanning in which the scanner does not scan all its targets simultaneously but separately. In other words, it uses more time to scan the whole target network in order to hide its activity better. However, we do not know if those flows are coming from the same source, at least not by using a single overview scatter plot. Thus, we select the red data points in the middle of the scatter plot in Figure 25 and visualize the selected sub-dataset from more aspects as shown in Figure 26, which meets the second design requirement: the local details drill-down.

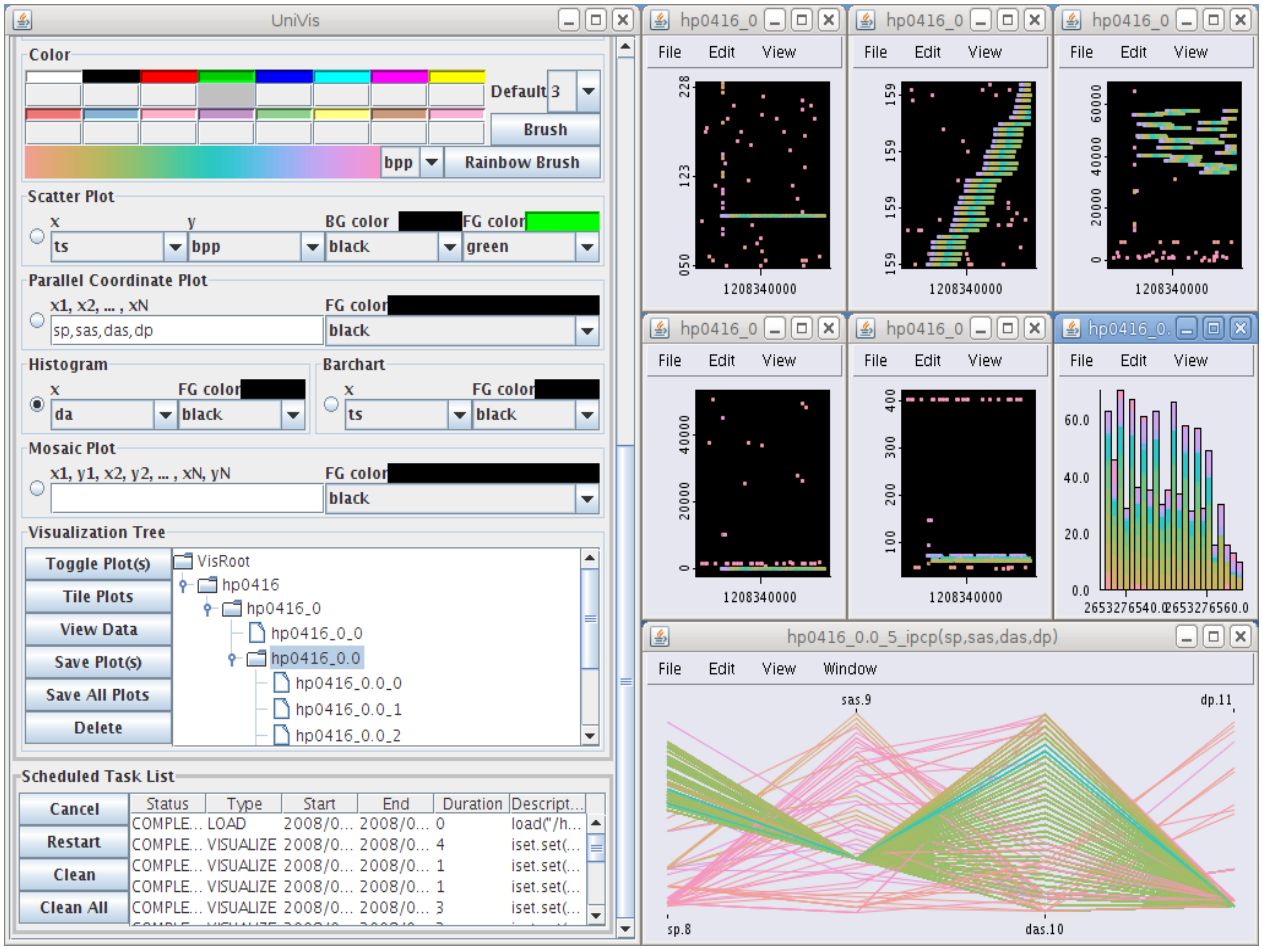


Figure 26: Using scenario details drill-down

Basically, we use other plotting sub-blocks in the “Interactive Visualization” block to request the plots we need. For the plots shown in Figure 26, from the left to the right and from the top to the bottom, we have first five scatter plots all using start time as x axis, and source IP address string, destination IP address string, source port number, destination port number and byte per packet respectively as the y axes. The sixth plot is the histogram of destination IP address string. The last plot is the parallel coordinate plot using source port number, source IP address string, destination IP address string and destination port number as parallel y axes sequentially. Because all plots linked to the same dataset, the third design requirement: the multi-view linking is fulfilled as well. The linkage is even more enhanced as the data points in all plots are brushed with the same color encoding scheme according to the values of the byte per packet attribute. The dark green color in the parallel coordinate plot clearly shows the fan out pattern, which means a single source IP address have traffics with a lot of destination IP addresses. The second scatter plot of “das vs ts” clearly shows the similar content pattern of the flows received by each destination IP address, because we assume that the byte per packet attribute can reflect the content to some extent. When we press the CTRL key and leave the mouse cursor over the line of the same color in the third scatter plot of “dp vs ts”, the detail-on-demand feature pops out a tooltip telling the

destination ports are all “21”. Based on the above evidences, it is not difficult to confirm a “creepy crawly” IP-sweep scanning is in action.

Finally, by selecting the “hp0416” node in the “Visualization Tree” block and clicking the “Save All Plots” button, the information of every data node and the image of every plot node are saved as text files and PNG image files respectively for later use. So the forth design requirement: the procedure extraction is also met.

6.7 Evaluation

In the previous section we briefly examine our implementation concerning the design requirements by studying a common use case. It turned out that all the four main design requirements are well satisfied. In this section our implementation of the UniVis prototype is evaluated more comprehensively against the criteria described in Section 1.3 with regards to the system design principles stated in Section 5.2.

For the functionality, the UniVis supports some of the most common used plotting schemes such as scatter plot, parallel coordinate plot, histogram and bar chart. It supports important interaction techniques such as linked selection, linked color brushing, and so on. Its ability in helping to find network anomalies is verified by a small use case. More discussions on this ability will be given after Chapter 7.

For the perception efficiency, the UniVis takes some human cognition knowledge and human perception principles into account in the design phase. For example, sensory symbols formed by shapes such as circle, box and lines are used in the visualization rather than arbitrary symbols. In such way more visual patterns may appear as pre-attentive features hence easier to be identified according to the Gestalt Laws. Also, the visualization tree is designed in order to keep a clear view of the structure of the visualization exploration procedure, in case too many plots are inevitably generated and seriously decrease the human perception speed by exceeding the limited visual working memory of human brain.

For the robustness of the visualization, the UniVis mainly solves it by the interaction techniques. For instance, the alpha-channel provided by the iPlots can be used to show the data density when most parts of the data overlap with each other. In addition, the details-on-demand features such as the tooltip display and the zooming avoid showing too much unnecessary details and distracting too much attentions. However, there are no specific techniques applied in the UniVis in order to prevent the intentional DoI attacks, mainly due to the insufficiency in this direction of research.

As for other common criteria, UniVis is designed to satisfy them as well as possible.

For the scalability, because the actual plotting process is carried out by the iPlots written in Java, the maximum amount of input data is between 500 000 and 1 000 000 flow records, depending on the processing power of the computer. This is not sufficient for the overview of the traffics measured on a backbone router, but good enough for exploring the local interesting spots.

For the extensibility, the modular design of the UniVis allows adding more functions rapidly by only implementing new scripts instead of modifying a lot of source codes. The whole component such as the iPlots can be upgraded or even be replaced independently.

Last but not the least, the UniVis is easy to use by its straightforward GUI control panel. Some personnels of the UNINETT CERT have tried it a bit and found they can use it almost immediately without much further instructions.

7 Experimentation

This section reports some of the investigations by using the UniVis in the routine NetFlow data measured by the routers managed by the UNINETT, focusing on the three malicious activities mentioned earlier in Section 3.3. Operational details of the investigations are omitted, as most of the exploration operations are already described in detail in the use case in Section 6.6.

7.1 Environment

7.1.1 Data Source

In general, all the NetFlow data used in the experiments are provided by the UNINETT CERT. Some of them are generated by the routers deployed on the backbone network, which is usually sampled due to the huge traffic throughput. We call such routers the backbone routers in this report. The other data are generated by the routers deployed on the edge between the smaller networks and the backbone network. We call them the border routers. Because those smaller networks usually have much smaller amount of traffics, the border routers are normally configured to generate unsampled data.

Besides the difference of the sample rate, the backbone routers and the border routers also have several other different configurations that are important to our experimentation. For instance, the reporting interval, flow inactive timeout, and flow maximum timeout, etc. We will specifically list those different configurations for each data we are using in the following experiments.

Because the packet analysis is our only method to identify the malicious hosts and the only way to confirm our experiment results, it is more convenient for our analysis to preserve their original IP addresses in the NetFlow data. Therefore, the NetFlow data used in the following experiments are not anonymized during the investigation, though they are hidden in this report for privacy reason.

With the above notices in mind, four NetFlow datasets are collected for analysis, namely the common dataset, the honeypot dataset, the IRC botnet dataset and the HTTP botnet dataset. These datasets are respectively generated by four routers deployed in four different observation points in the network of UNINETT, as illustrated in Figure 27. More details of the datasets are given in Table 7.1.

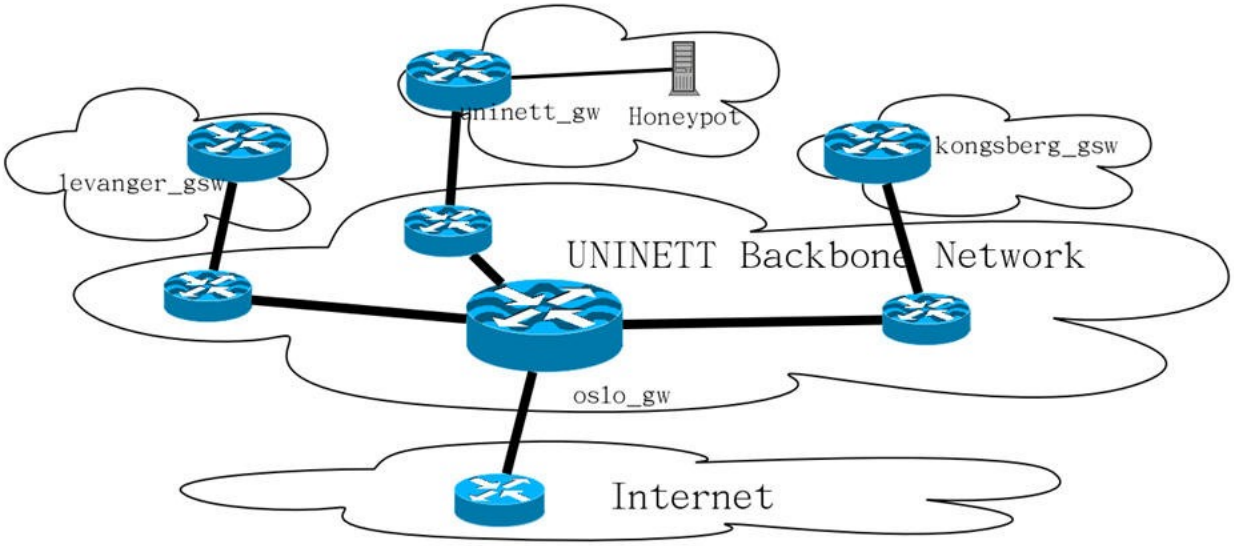


Figure 27: Observation points of the datasets for experimentation

Purpose	Common	Honeypot	IRC Botnet	HTTP Botnet
Router	oslo_gw	uninett_gw	kongsberg_gsw	levanger_gsw
Time Window	2008-04-09 2008-04-16	2008-04-11 2008-04-16	2008-04-11 2008-04-15	2008-03-30 2008-04-15
Flow Number	319 912 003	320 862	301 445 738	40 562 332
Sample Rate	1:100	1:1	1:1	1:1
Inactive Timeout	15"	15"	15"	15"
Active Timeout	1'	30'	30'	30'
Report Interval	15'	15'	5'	5'

Table 7.1: Details of datasets from four data sources

- Common
  This dataset contains the sampled traffics observed by a main backbone router located at Oslo, namely “oslo_gw” as shown in Figure 27. As a main gateway, most of the traffics flowing between the UNINETT network and the external Internet can be observed here. Such traffics can be internal traffics following close loops within the UNINETT networks, external traffics bypassing UNINETT networks, and also the traffics between the UNINETT internal networks and the external Internet. So, it is one of the best observing points within the backbone networks. In other words, this dataset has a good chance to contain the malicious traffics contained in the following datasets, which are observed by the border routers.
  Therefore, this dataset is mainly used for the second part of the experimentation, which according to our methodology, is to verify the effectiveness of our results attained in the

first part of the experimentation using the following datasets.

- Honeypot
  This dataset contains the unsampled traffics observed by a border router named "uninett_gw", which serves as the gateway between the Internet and the local network used by the UNINETT itself. The dataset is called honeypot data because it is produced by filtering the traffics only relevant to the subnet (net 158.*.*.*/24) allocated to a honeypot facility set up by the UNINETT CERT.
  Because a honeypot is configured to a certain security level that is vulnerable enough to download the malwares but still secure enough to prevent them from been executed, this dataset should contains a lot of scanning traffics and possibly some C&C link traffics.
- IRC Botnet and HTTP Botnet
  The IRC botnet dataset and the HTTP botnet dataset contain the unsampled traffics observed by the border routers of two campus networks respectively. These two border routers are chosen because the UNINETT CERT confirms by packet analysis that some computers within these two networks are compromised bots and actually communicate with some known botnet C&C controllers on the Internet. For the IRC botnet dataset, the IP addresses of the C&C controllers are 67.*.*.* and 84.*.*.*. Both of them use TCP port 3306 to establish C&C links with the bot 158.*.*.*. As for the HTTP botnet dataset, the IP address of the C&C controller is 64.*.*.*, which uses TCP port 80 to communicate with another bot 158.*.*.*.

7.1.2 Additional Facilities

As mentioned in the description of the honeypot dataset, the UNINETT CERT has established facilities such as the honeypot for assisting the security management of their networks. In addition, the packet analysis is clearly not supported by the NetFlow data but pcap data.

Besides the honeypot facility, a mechanism described by Knutsen in [46] is also very useful in our work. This mechanism is mainly used to redirect the traffics relevant to the known botnet controllers to a sinkhole, where the traffic packets are finally dropped instead of received by the destined receiver. Hence, this mechanism cuts off the C&C links between the botnet controllers and the bots. In our case, we can stop such rerouting mechanism for a couple of botnet controllers for a short period, in order to find the bots. As a matter of fact, this is how we get the IRC botnet dataset.

7.1.3 Computing Resources

The main information related to the processing power of the computer running the UniVis system is given in the following list. Notice that it is a common desktop personal computer with enhanced processing power rather than any specialized workstation with instinct high performance.

- CPU core number: 4
- CPU frequency: 2 333.509 MHz
- CPU cache size: 4 096 KB
- CPU bogomips: 4 666.96

- Memory: 5 189 696 KB

Most of the time the UniVis system ran smoothly during the experimentation on the above computer. However, When the input flow number is higher than 100 000, some of the visualization functions such as parallel coordinate plotting slow down rapidly. When the input flow number is higher than 1 000 000, most visualization functions are too slow for interaction.

7.2 Experimentation planning

According to our methodology, the whole experimentation is organized in a hierarchical structure and carried out in an iterative manner, as depicted in Figure 28 and explained as follows.

Firstly, several typical types of the malicious network activities are selected. In this experimentation two of the three malicious activities introduced in Section 3.3 are chosen except for the DoS attack. Because each of these activities should have certain characteristics, which may help to distinguish them from each other and most importantly, from the normal activities, we then need to use several visualization methods to explore such characteristics in the given datasets. However, it is usually difficult to analyze a complex dataset as a whole while its natural subsets, if any, are not well examined and understood yet. As we know, each flow record in the NetFlow data are consisted of several attributes. In addition, because the processing speed of our visualization system depends on the amount of the input data, the preliminary breakdown of the dataset in a sense gives a considerable improvement in the performance of the system.

Hence secondly, for each type of malicious activity, the subsets of the original unsampled data are extracted according to the selected attributes and the combinations of those attributes. The selection and the combination of the attribute are mainly based on the preliminary knowledge of the malicious activity. Unsampled data are used first simply because it contains more complete information then the sampled ones.

Thirdly, when the dataset is prepared, they are visualized for the first round by all the visualization methods available in our system, namely the scatter plot, the parallel coordinate plot, the histogram and the bar chart. As a result, some plots may show clear visual patterns while some others may not. Further more, some visual patterns may either correspond to actual malicious activities or not. Thus, analysis of the observations are followed, trying to explain the reasons of the result and to give suggestions that may help to improve the results for the next step.

Fourthly, with the initial visualization results and analysis, the strength and weakness of each visualization method applied to the specific malicious activity are exposed and compared between each other. The comparisons result in a mixed way of usage of several visualization methods together to explore the specific malicious activity. Interaction techniques such as linked selection and color brushing are used in attempt to glue these visualization methods together into an organic one. Such synthesized procedures are employed in the second round of experimentation over the same data, followed by analysis as in the third step. This step can be performed iteratively for several times since the experiences gained each time of using the interaction techniques may help to have new discoveries and hence improve the results.

Lastly, the empirical procedures of the interactive visual exploration gained from examining the unsampled datasets are applied to the sampled common dataset. As the sampling reduces the

amount of input data to our system and thus increase the processing speed, it is definitely worth knowing if such reduction will also affect the previous results over unsampled data or not. And if it does, how badly is the degradation. Though the experiments on the sampled data are carried out separately from those on the unsampled data, the result and analysis are reported together following the above steps.

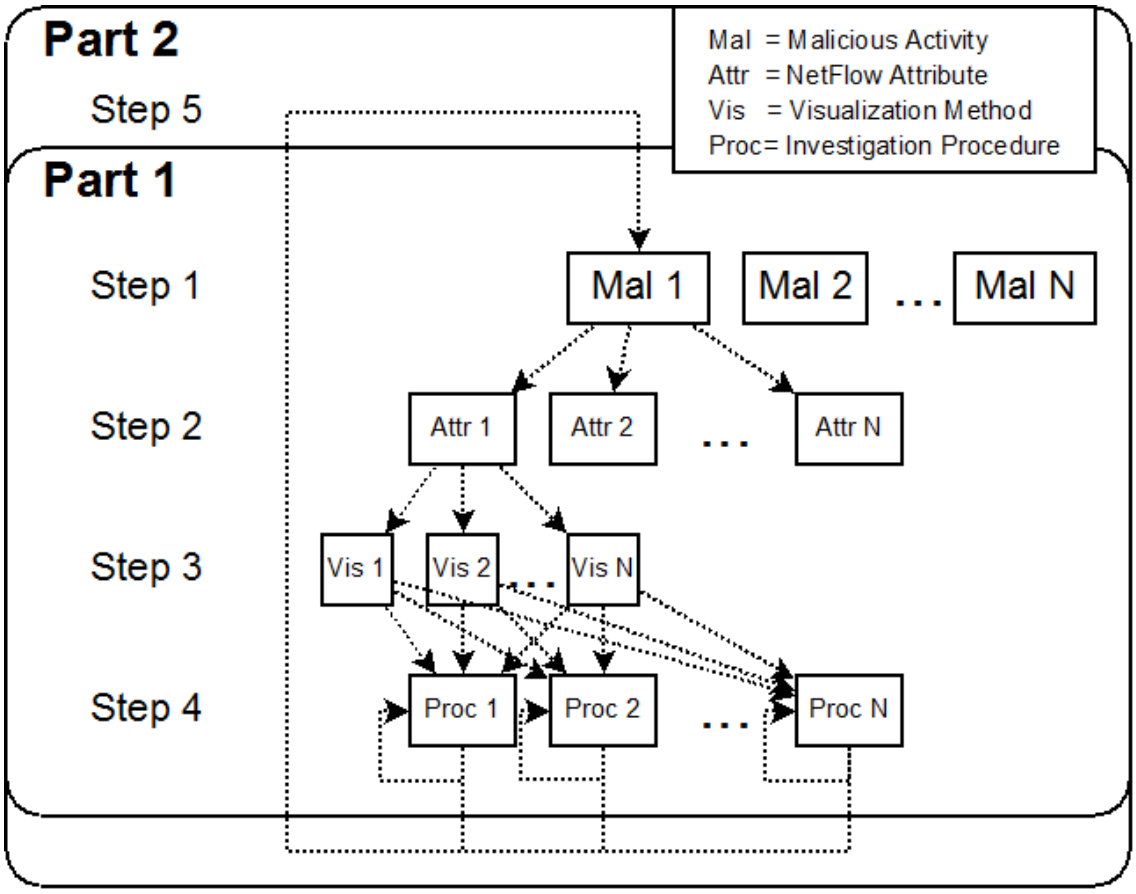


Figure 28: Experimentation planning

From step 1 to step 4 the datasets are unsampled attained from specific observation points where the traffics of specific malicious activities are for sure captured. These steps form the first part of our experimentation, where the features of malicious activity are discovered and the procedures to identify such activities are logged. In step 5, these features and procedures are applied to the sampled common dataset in order to test their effectiveness in the identification of the corresponding malicious activities. Step 5 is the second part of our experimentation. These two parts correspond to the two parts of our methodology of detection of malicious network activity.

7.3 Scanning

As mentioned in Section 7.1.1, the honeypot dataset is used in the first part of this experimentation. Though The whole dataset is used during the experimentation, in this report

only the experiments using the dataset captured on 2008-04-16 are described as an example. According to the discussions on the scanning given previously in Section 3.3, the basic traffic phenomenon we look for here are the IP-sweep scanning, the port-sweep scanning, and the creepy-crawly scanning.

7.3.1 NetFlow Property Selection

Because the scanning is about the communication relationships between network hosts, NetFlow properties relevant to the IP addresses and port number are crucial, such as the source IP address (sa/sas), destination IP address (da/das), and the source and the destination port number (sp, dp).

In addition, most scannings have strong bounds with time. That is to say, time relevant attributes are useful as well, such as start time (ts/tss), end time (te/tes) and duration (td).

Moreover, most scanning traffics sent to different destinations have the same or very similar content. Though we do not record the traffic content directly, the content related attributes such as the byte number (byt), the packet number (pkt) and the byte per packet (bpp) may also reflect the phenomenon.

As for the combination of the attributes, we mainly focus on the combination of two attributes. This is because the combination of more attributes require parallel coordinate plot, which is much more resource consuming than the two dimensional scatter plot.

One type of the combination of two attributes is to fix the time relevant attributes and replace other attributes in turn. This type can be called the temporal combination. If we fix the content relevant attributes and replace other attributes in turn, the resulting combinations can be called the payload combinations. When we combine the IP address or port number attributes, the resulting combinations can be called the connection combinations.

The data of each attribute or combination of several attributes suggest different aspects of the traffic. Thus good visualization should reveal the same aspects and accentuate their characteristics.

7.3.2 Investigation

The IP addresses allocated to our honeypot is a subnet 158.*.*.* with 24 bits netmask. Thus, the scannings we look for are from the Internet towards this subnet. The traffics going towards the honeyhot are called ingress flows which are mostly malicious, while the traffics on the other direction are called egress flows. Besides the basic plotting schemes, color brushing, linked selection, details-on-demand are the main interaction techniques.

Especially for the color brushing, both the filter brushing and the rainbow brushing are used. The filter brushing are mainly used to distinguish the direction of flows. For instance, taking one end point of the communication as the original end point, hence the flows going towards it are ingress flows and the flows going away from it are egress flows. In this report, we usually brush the egress flows as blue and the ingress flows as yellow. The rainbow brushing are in general used to show the value distribution of an additional attribute to the original plotting. For example, when a rainbow brushing according to the value of attribute bpp is applied, the flow data with different

bpp values can be easily distinguished by their colors.

Only some key visualizations are presented in this section, though much more plots has been created and examined during the experimentation.

Global Overview Snapshot

According to the preliminary knowledge of scanning, several scatter plots are used to show the main characteristics of scanning. We denote the types of scatter plot by the attributes of data assigned to its two axes. For example, scatter plot “A vs B” denotes a scatter plot with its y axis representing attribute A and with its x axis representing attribute B. The following scatter plots are all of type “das vs ts” in order to identify the IP-sweep scanning and the IP based creepy-crawly scanning. The investigation of the port-sweep scanning is omitted because they can be identified by the same principles in the “dp vs ts” scatter plot.

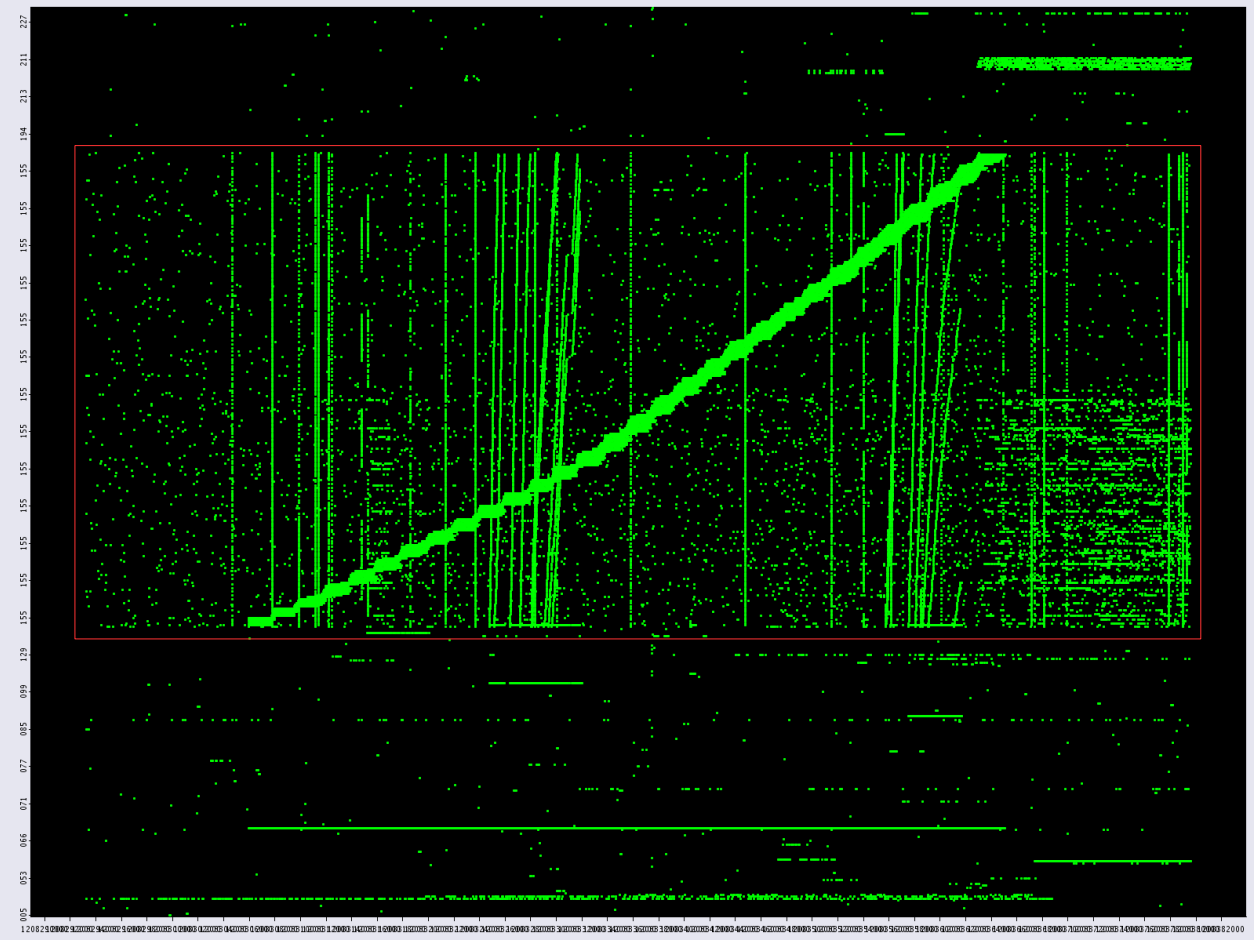


Figure 29: Original scatter plot "das vs ts"

Figure 29 shows the temporal pattern of destination IP addresses. A preliminary observation is that the data points in general form lines in the plot. Horizontal lines suggest that many flows go to a single IP address, which is a common characteristic of a server. The vertical lines indicate

that during a very short period of time many flows go to multiple IP addresses, which is one important characteristic of scanning traffic and worth more examinations. In addition, the data points inside the red box are traffics flowing towards the honeypot. Thus we should look more into the vertical lines within that red box in order to see if they fits other basic characteristics of scanning.

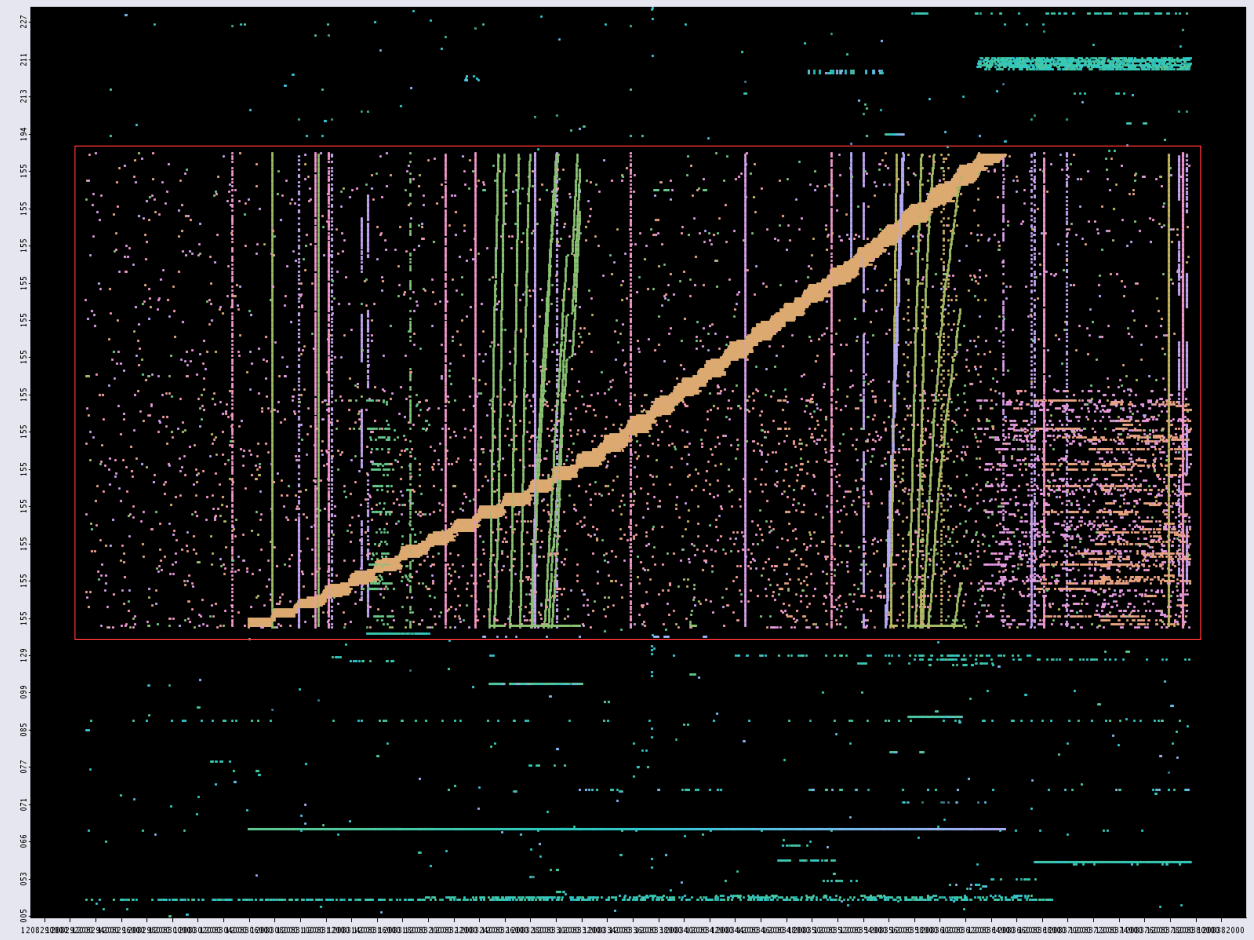


Figure 30: Scatter plot "das vs ts" with rainbow brushing by sas

By brushing the original “das vs ts” scatter plot with the rainbow colors according to the sas attribute we get Figure 30. Flows coming from different source IP addresses are brushed with different colors. As can be easily observed, the data points forming each vertical line are of the same color, which suggests that at the same time a single host or a small group of hosts are sending flows towards many other hosts. After confirming that each of the flows forming the vertical lines actually comes from very few hosts, the only basic characteristic left is the similarity of flow payloads.

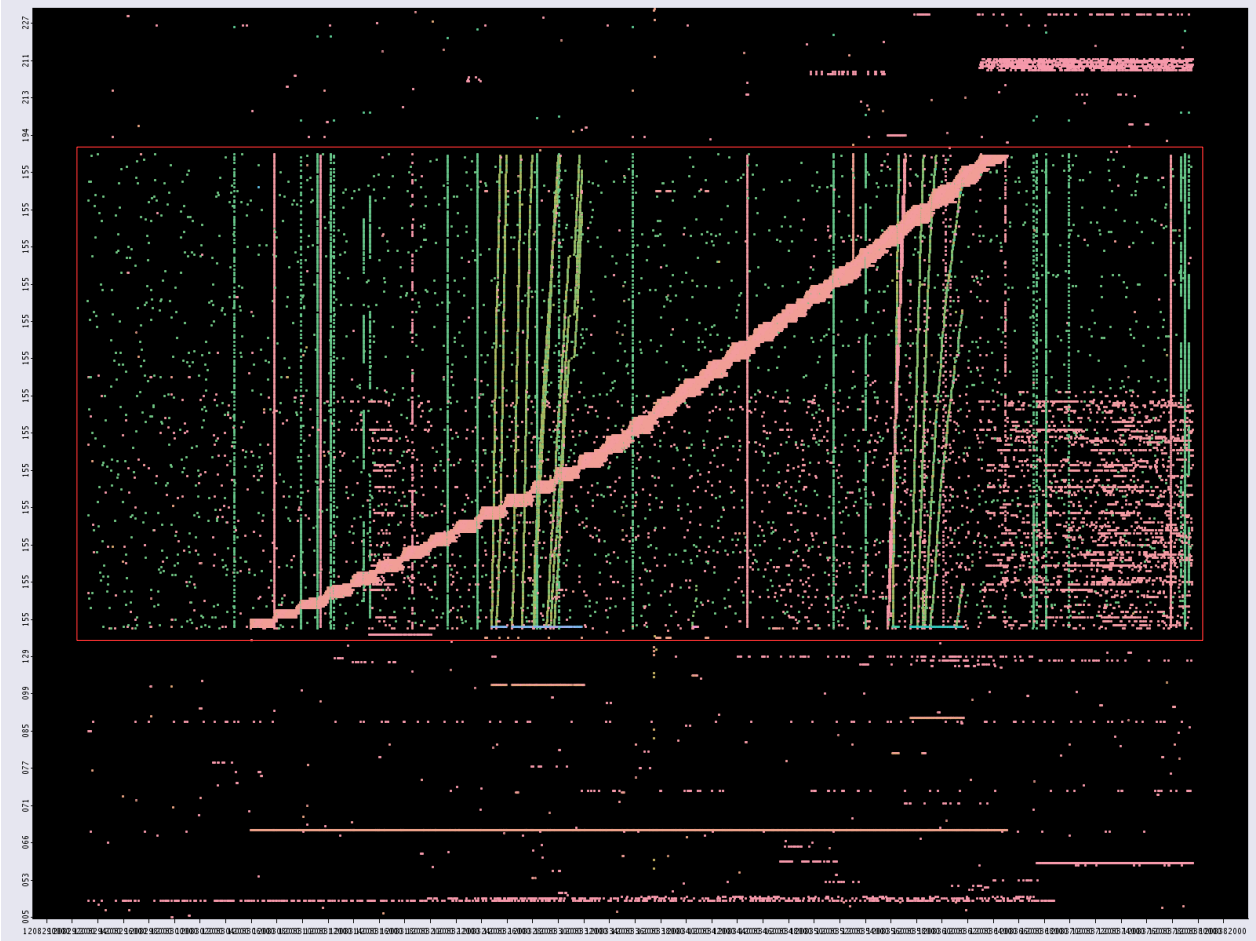


Figure 31: Scatter plot "das vs ts" with rainbow brushing by bpp

As discussed before, the only attributes relevant to the payload is byt, pkt and bpp, among which bpp is reckoned as the best one to represent the characteristics of the payload. Therefore we get Figure 31 by brushing the original plot with the rainbow colors according to the bpp attribute. Again we can observe the consistent color of the data points forming each vertical line, suggesting very identical payload in each flows. Hence up to now we can confirm that those vertical lines are actual IP-sweep scannings. Because the thick diagonal line lasts for a longer time period, it can be identified as IP based creepy-crawly scanning. Of course they can also be port-sweep scannings depends on result of the same investigation into the scatter plot of “dp vs ts” generated from the same dataset.

Local Detail Drill-down

Most of the time, we need to know more details of the malicious activity, which can not be efficiently and precisely acquired from the previous global overview snapshots. For example, from only a single scatter plot of “das vs ts” color encoded by an additional attribute, we can not conveniently find out the exact value of the attributes. At this moment, we need to drill down to

the interesting spots for more detailed information. Two examples of local detail drill-down are given as follows for the IP-sweep scanning and the creepy-crawly scanning respectively.

- **IP-sweep**

  In Figure 32, two vertical scanning lines are selected as our examples of the IP-sweep scanning, as marked by the red color. Apparently they are slightly different since the right scanning line has continuous destination IP addresses while the left scanning line is not but seems to have constant gaps of missing destination IP addresses.

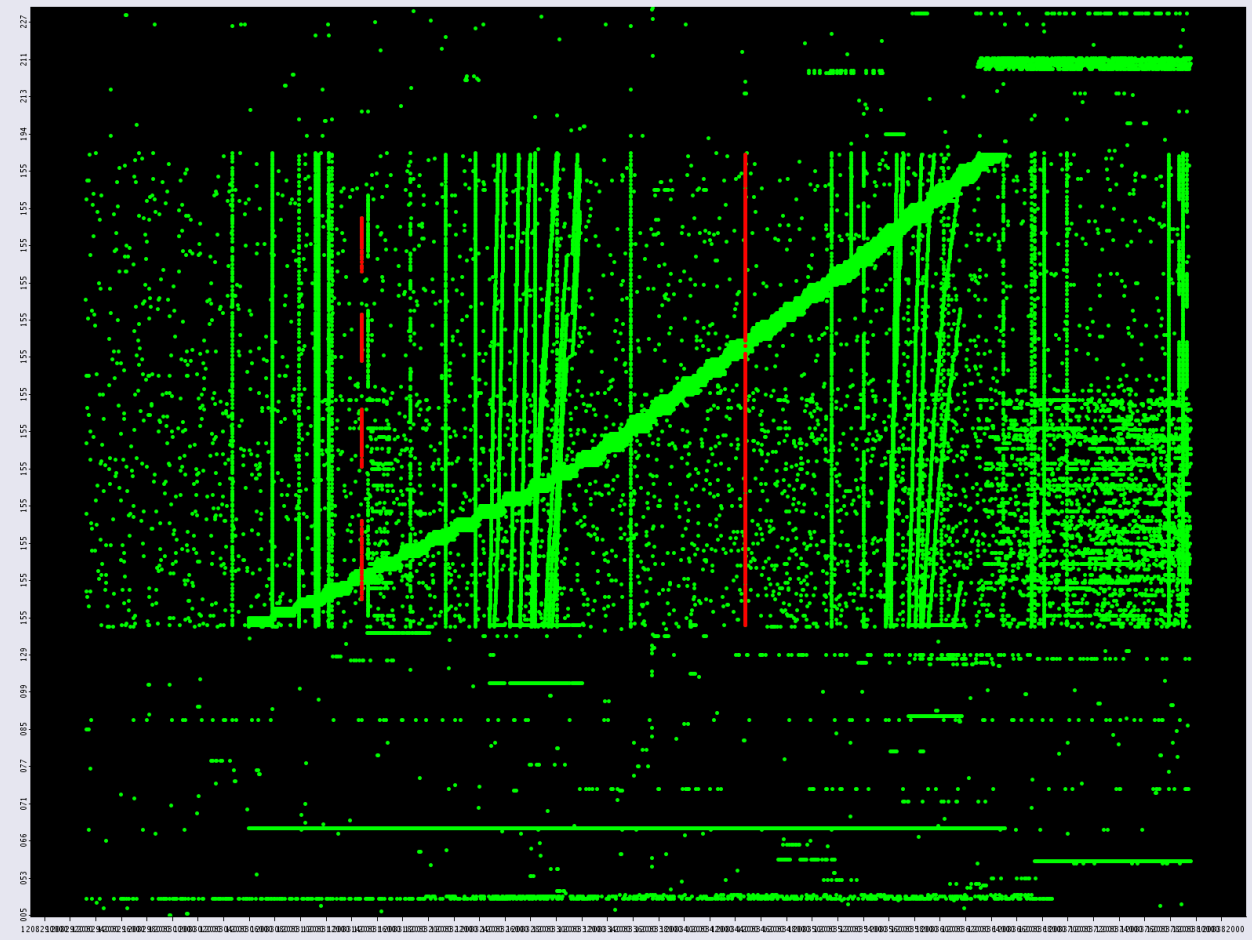


Figure 32: Selection of IP-sweep scanning from overview snapshot

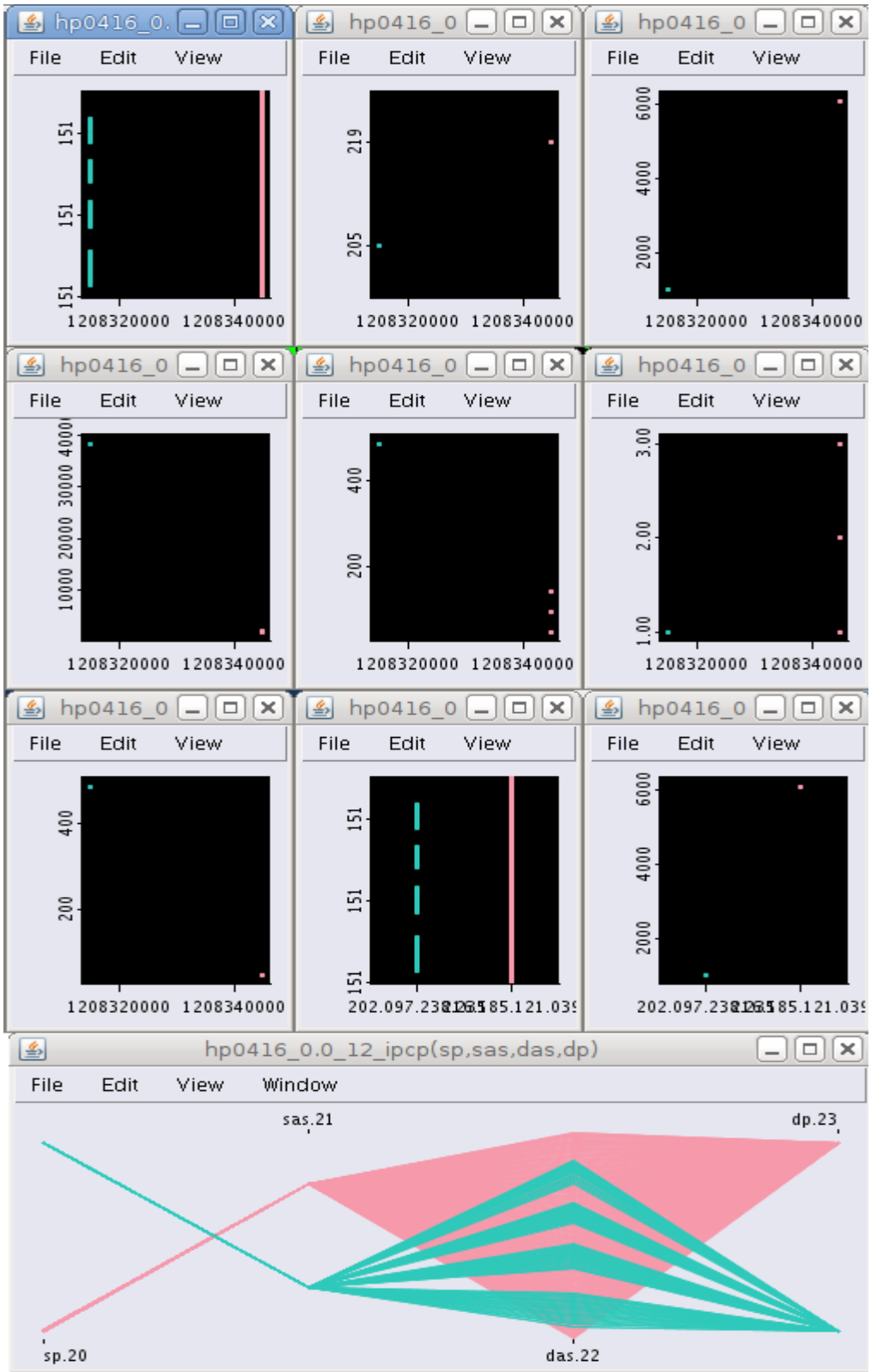


Figure 33: Local detail drill-down of IP-sweep scanning

In Figure 33 we sequentially plot nine scatter plots of “das vs ts”, “sas vs ts”, “dp vs ts”, “sp vs ts”, “byt vs ts”, “pkt vs ts”, “bpp vs ts”, “das vs sas”, “dp vs sas”, and finally one parallel coordinate plot of “sp vs sas vs das vs dp”, from the left to the right and from the top to the bottom. All these plots are brushed with the rainbow colors according to the bpp attribute. Each plot gives an aspect of details of the scanning. Hence we read the information for the two IP-sweep scannings as given in Table 7.2.

Group	Source IP Address	Destination Port	Source Port	Byte Number	Packet Number	Byte per Packet	Flow Number
Blue	202.*.*.*	1026 1027	38275	485	1	485	127
Pink	216.*.*.*	6080	1949 2102 2103 2202	144 96 48	3 2 1	48	254

Table 7.2: Details of IP-sweep scanning

- **Creepy-crawly**

  In Figure 34, the thick diagonal scanning line is selected as our examples of creepy-crawly scanning, as highlighted by the red color again. This scanning in all lasted for approximately 8 hours and swept the whole IP address space of the honeypot. A closer look shows that the whole scanning period can be evenly divided into many small time slices, in each of which a small group of continuous IP addresses are scanned. In each subsequent time slice after the previous one, the target IP addresses increase linearly.

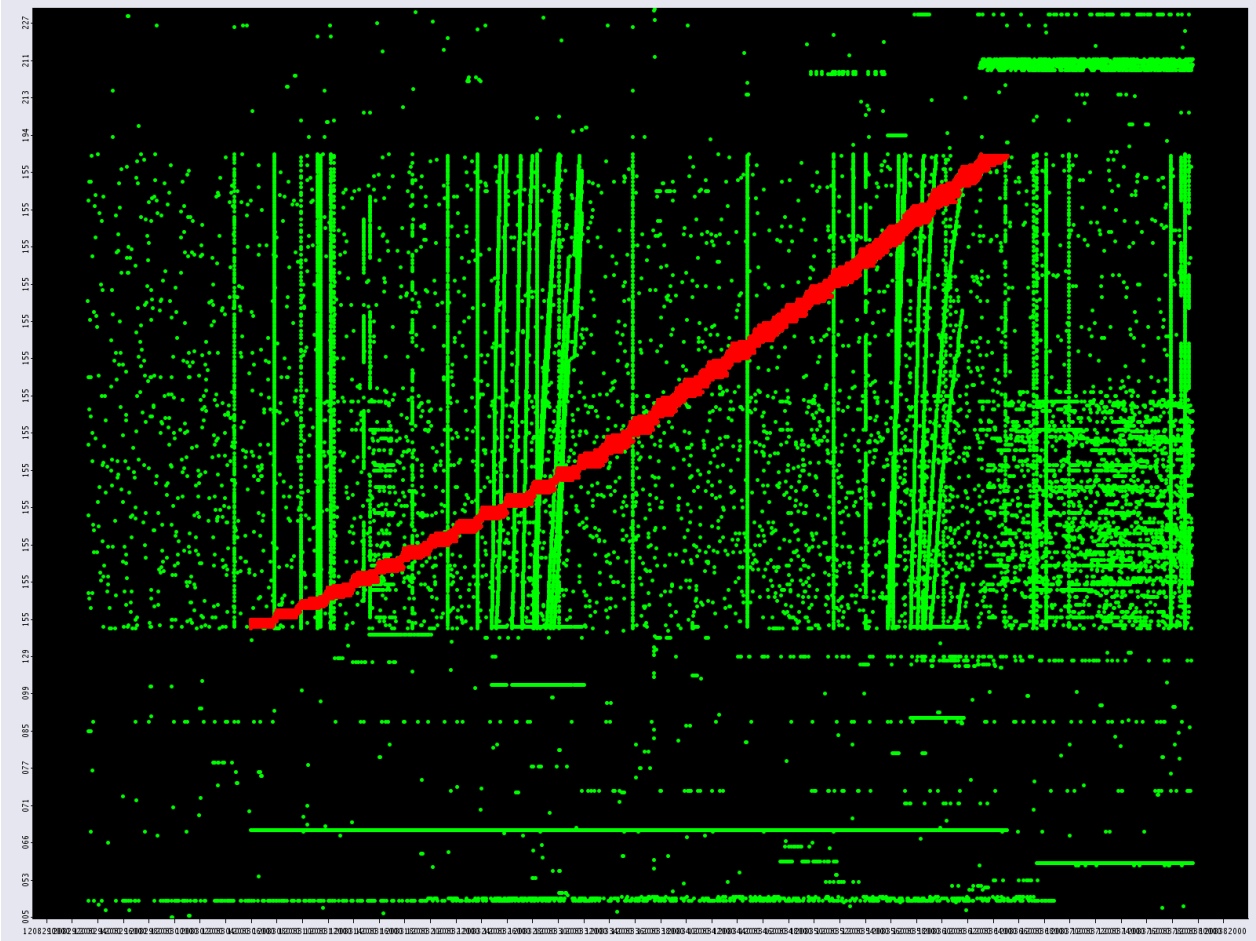


Figure 34: Selection of creepy-crawly scanning from overview snapshot

In Figure 35 we perform the same visualizations as we did for the IP-sweep scanning. In the same way, we hence read the information for the creepy-crawly scanning as given in Table 7.3.

Source IP Address	Destination Port	Source Port	Byte Number	Packet Number	Byte per Packet	Flow Number
66.*.*.*	21	32829-60999	52-37532	1-529	52-72	7888

Table 7.3: Details of creepy-crawly scanning

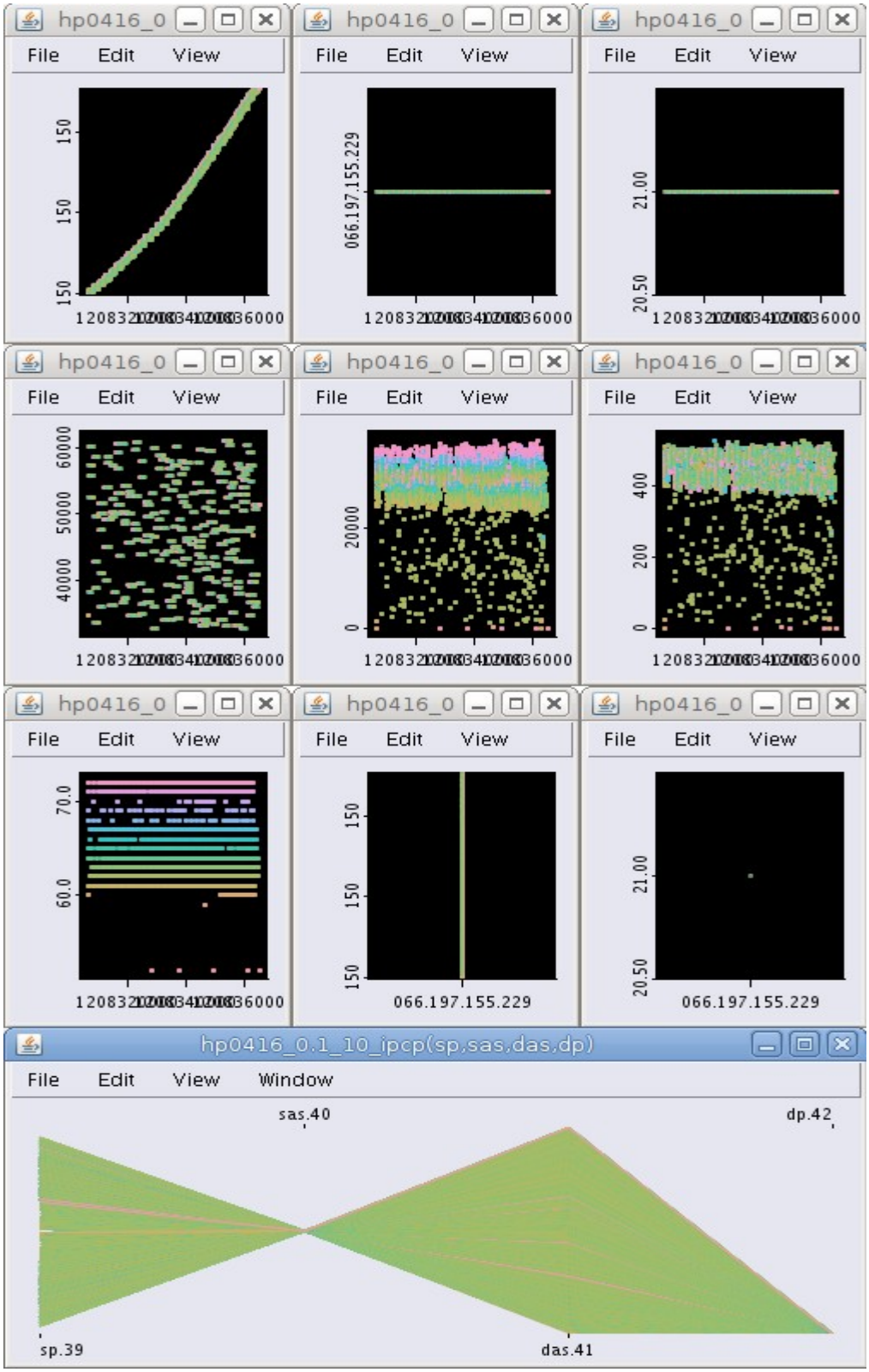


Figure 35: Local detail drill-down of creepy-crawly scanning

In addition to the information shown in Table 7.3, some of the drill-down visualizations give more information by their visual patterns than the more precise value readings of the data points.

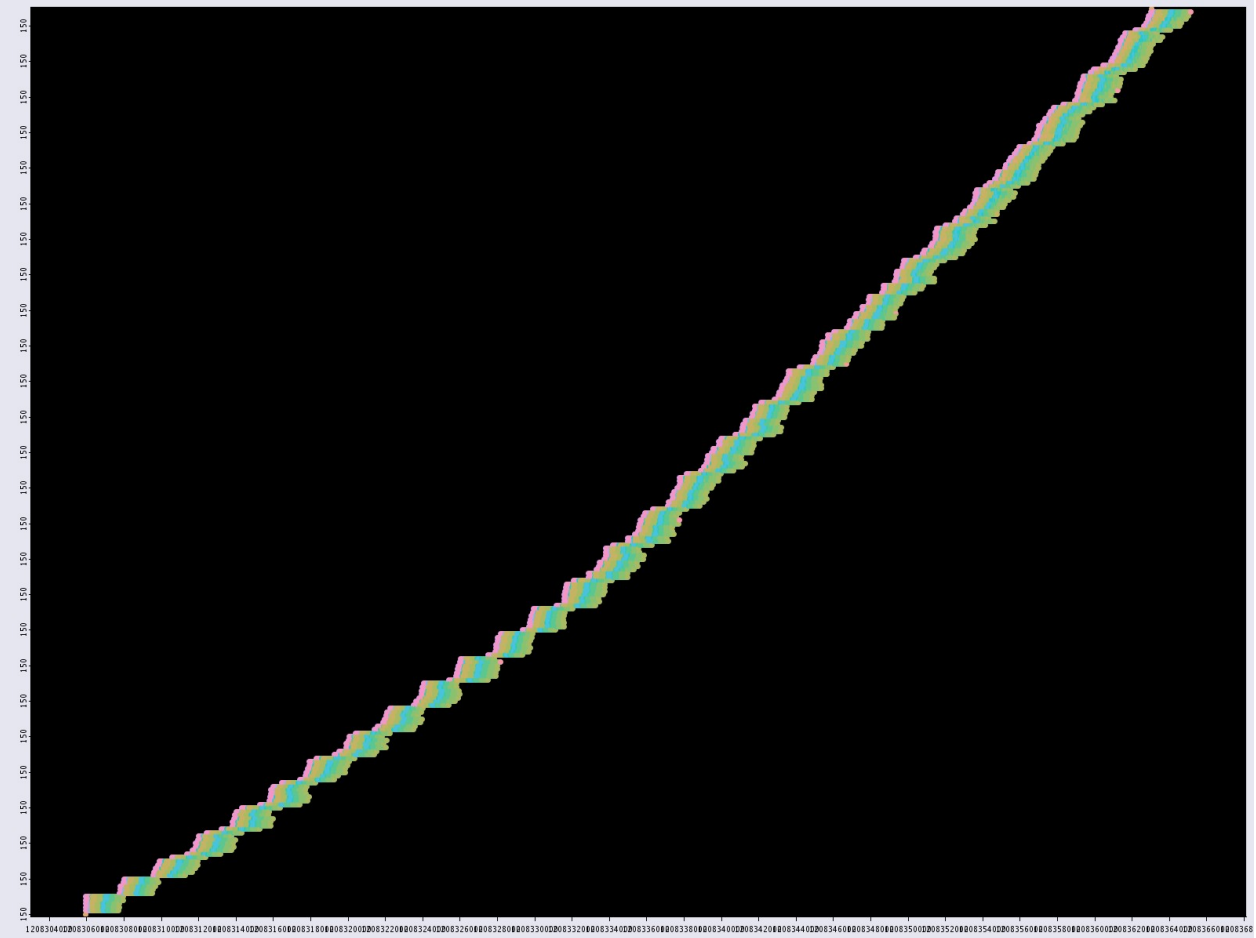


Figure 36: Local detail drill-down scatter plot of "das vs ts" for creepy-crawly scanning

In Figure 36, the local detail drill-down scatter plot of “das vs ts” rainbow brushed by bpp clearly shows that the flows for each target host have very similar payload pattern. More specifically, each horizontal line of the flows has a color spectrum sequentially and coarsely composed of pink, brown, green and then brown again. Such pattern of the payload can lead to two results. First, it further proves that this is a scanning. Second, it reveals that there are probably interactions between the scanner and the victims. As other plots shows the destination port of the scanning is 21 over the TCP protocol, which is the FTP service port, it is possilbe that the scanner is trying some attacks already such as the password guessing. Figure 37 of the local detail drill-down scatter plot of “sp vs ts” clearly shows the same color pattern of each line of flows as previously observed in Figure 36, which further confirm the conclusions gained from Figure 36. Such pattern does not appear in the plot produced in the same manner in the overview snapshot such as Figure 31 because the value range of bpp of the creepy-crawly scanning flows is too narrow. When the value range of bpp is much larger in the overview, the scanning flows can only be encoded with very similar colors in the rainbow color space.

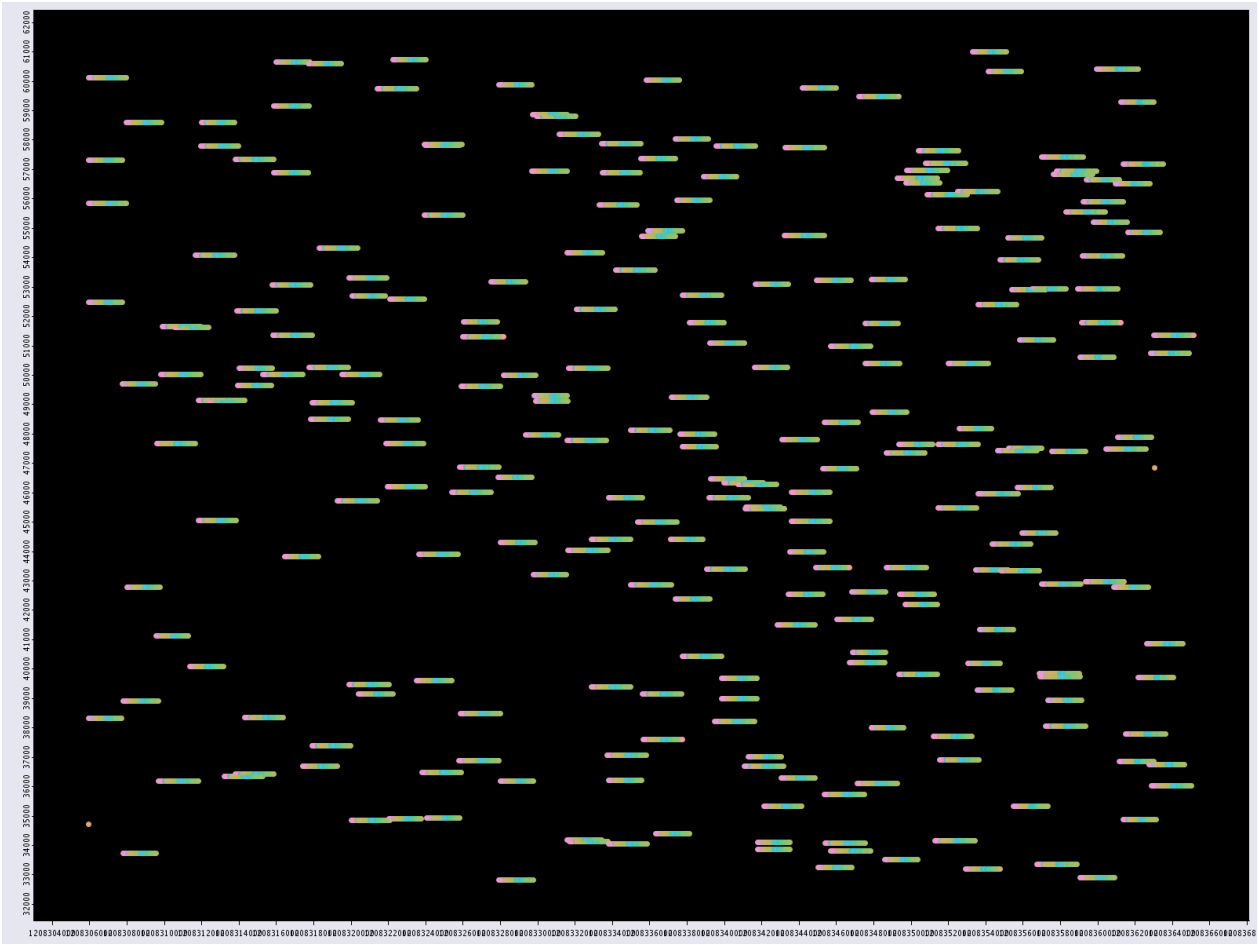


Figure 37: Local detail drill-down scatter plot of "sp vs ts" for creepy-crawly scanning

7.3.3 Application on Common Dataset

After the close examination of the honeypot dataset, here we try to find the same visual pattern of the malicious activities in the common dataset. As mentioned earlier, the common dataset is gathered from a different observation point of the network with sampling enabled and covers the same time window.

Figure 38 gives the overview snapshot of the honeypot relevant traffics. Such overview is produced by the exact same way as the one used to generate Figure 31. That is to say, it is a scatter plot of “das vs ts” brushed by the rainbow colors according to the bpp attribute. The red box highlights the flows going towards our honeypot.

Compared to Figure 31, the flows are much less in Figure 38. The two vertical scanning lines which are used as our IP-sweep scanning examples totally disapear, so do many other vertical scanning lines. On the other hand, the thick diagonal scanning line of the creepy-crawly scanning still remains very clear and complete.

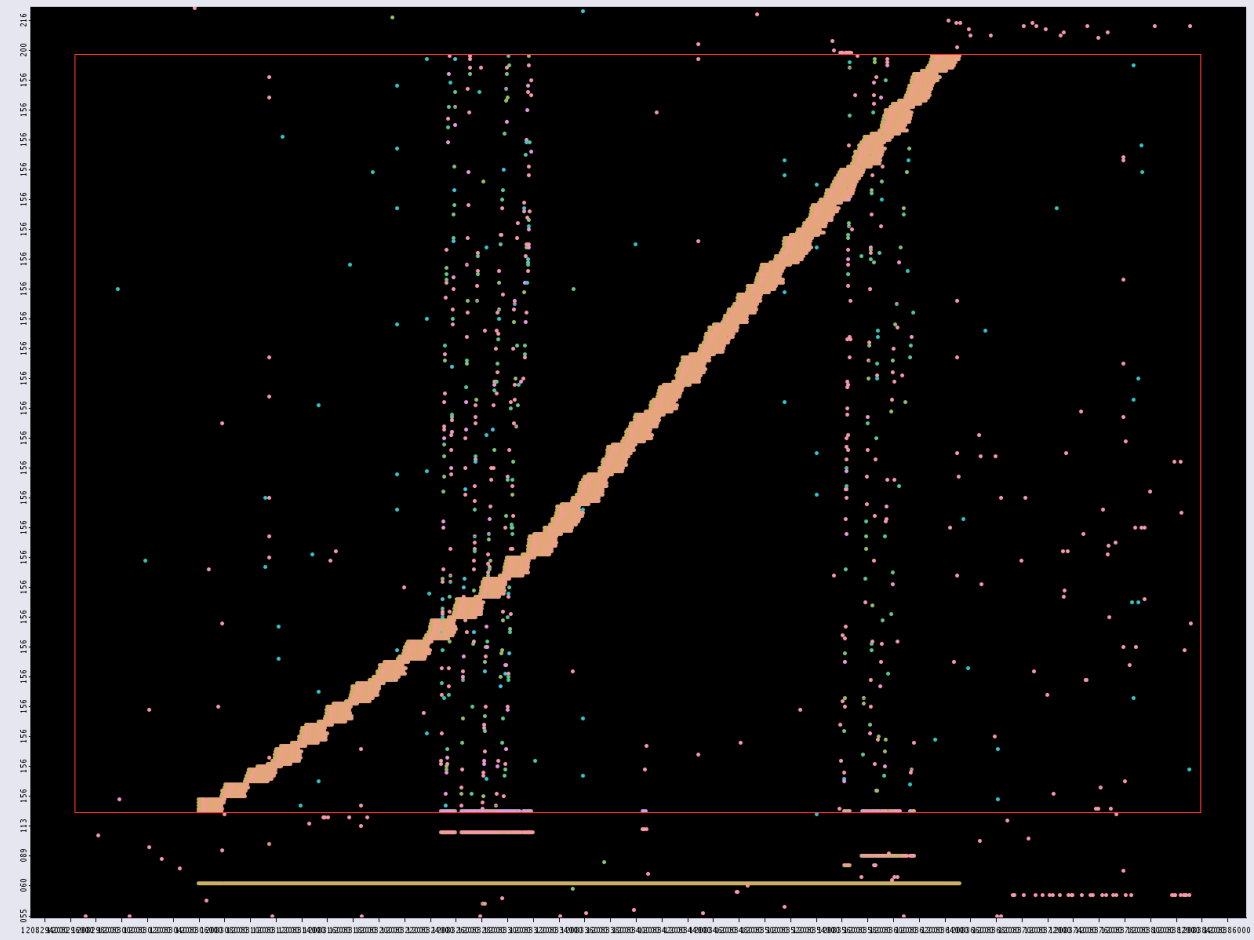


Figure 38: Scatter plot of "das vs ts" with rainbow brushing by bpp

The results of the two types of scanning are so different at least due to the following two reasons. First, because the common dataset is gained from a different observation point, it is possible that the traffics of the IP-sweep scanning never flow through this observation point. Second, assuming the traffics of the IP-sweep scanning all flow through this observation point, the packet number for each flow is too low to pass the sampling processing. As shown in Table 7.2, each flow have at most 3 packets in all. Among the massive other traffics, their chances of being noticed after the sampling of 1:100 sample rate are extremely small.

7.4 IRC Botnet C&C Link

As mentioned in Section 7.1.1, the IRC botnet dataset is used in the first part of this experimentation. Filtered by the known IP addresses of the controllers and bots, the dataset focuses on the known C&C links. Based on the discussion on the IRC C&C link in Section 3.3, the main traffic phenomenon we look for here are the initial malware downloading and the keep-alive signaling, though the malware downloading can not be identified in the end.

7.4.1 NetFlow Property Selection

The selection here still refers to the selection in Section 7.3.1. In other words, the temporal, payload and connection relevant attributes are all included in this selection, so are their combinations.

7.4.2 Investigation

Global Overview Snapshot

Because the traffics of the C&C links are usually very rare, we can not identify them directly by our current visualization techniques from any of the global overview snapshot. However, because we already know some C&C links, we can use them to filter out all the other traffics in order to focus on the C&C traffics for local details drill-down.

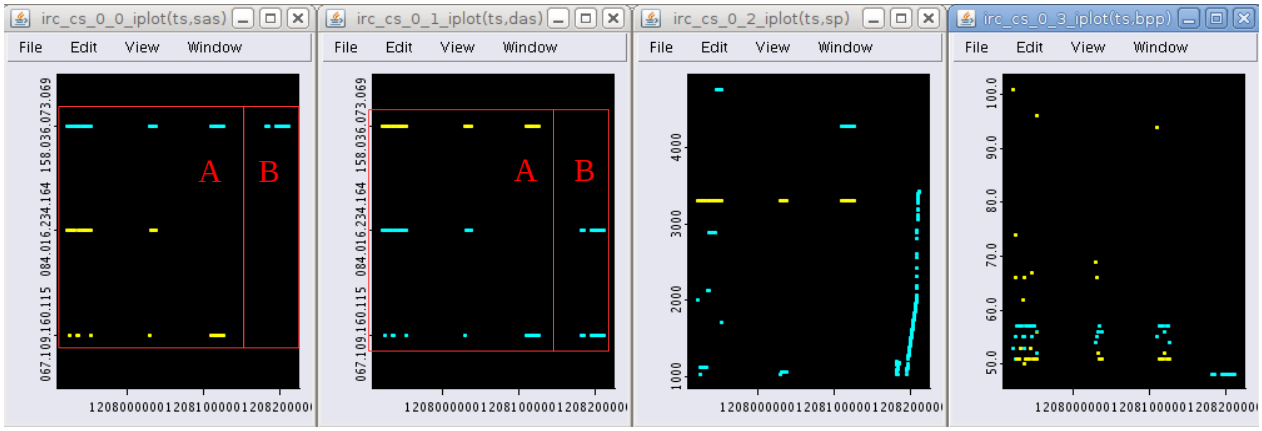
As introduced before, the IP addresses of the known C&C controllers are 67.*.*.* and 84.*.*.*. Both of them use TCP port 3306 to establish C&C links with the infected bot 158.*.*.*. In order to tell the direction of the flows in the plot when necessary, we denote the flows going from the controllers to the bots as ingress flows, and the flows of the other direction as egress flows.

Local Detail Drill-down

By four filter expressions composed of the port number, the source IP address and the destination IP address, we generate four sets of sub-datasets to investigate. In short, the first dataset filtered by both the controller and the bot IP addresses is for the extraction of the original visual patterns of the IRC botnet C&C links. Then, the other three datasets are filtered out by the filter expressions each removing one aspect of the initial full expression, in order to find more C&C links. Each dataset is visualized by four kinds of scatter plot, namely “sas vs ts”, “das vs ts”, “sp vs ts”, and “bpp vs ts”. All plots of each dataset are brushed by three color schemes: 1) the filter brushing according to the flow direction, whereby the ingress flows are blue and the egress flows are yellow. 2) the rainbow brushing according to the sp attribute, and 3) the rainbow brushing according to the bpp attribute.

- Filter Controllers and Bots

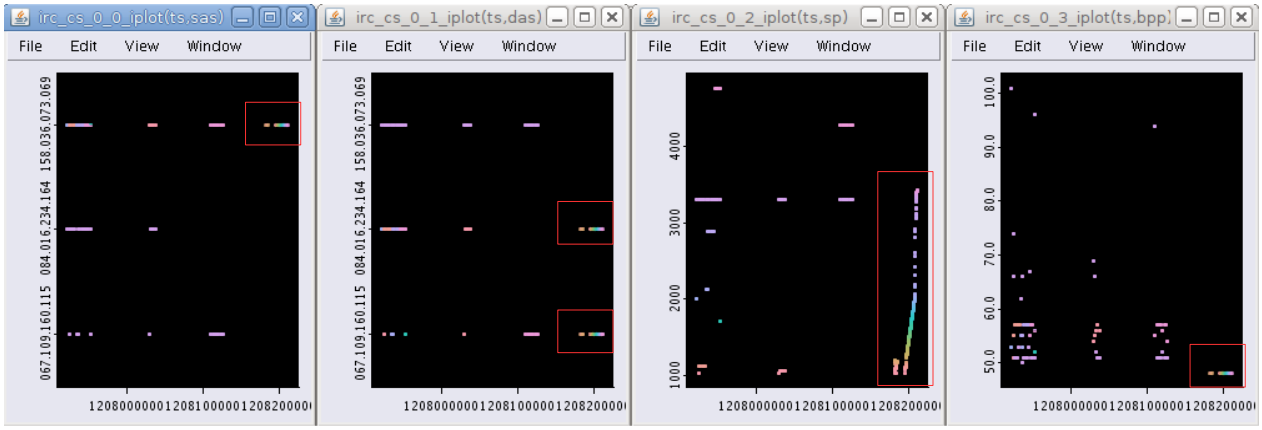
Figure 39 shows the plots brushed according to the flow direction. In Figure 39 (a) and (b) three lines formed by the data points with steady interval of 30 minutes, indicating that the two controllers and one bot have periodical outgoing or incoming communications. The same traffic interval indicates they are very likely to be of the same botnet. Notice the traffics are bidirectional in region A and become unidirectional in region B. The bidirectional traffics are considered common keep-alive IRC C&C links. The unidirectional traffics are caused by the isolation of the controllers. As a result the bot keeps contacting the controllers much frequently (every 180 seconds) than before, without any responding flows from the controllers. In Figure 39 (c) the unidirectional connecting attempts can be confirmed by the rapid increase of the port number on the port, which is a common behavior of the OS. Figure 39 (d) shows that the bpp values of the payload of each flow direction is quite stable most of the time. This characteristic is very similar to the keep-alive signals. In addition, the bpp value of the egress flows of the bot after the isolation of the controllers differs from the one before the isolation.



(a) (b) (c) (d)

Figure 39: Scatter plots of data filtered by controllers and bots, color brushed by flow direction. Plots (a), (b), (c) and (d) are scatter plots of "sas vs ts", "das vs ts", "sp vs ts" and "bpp vs ts" respectively.

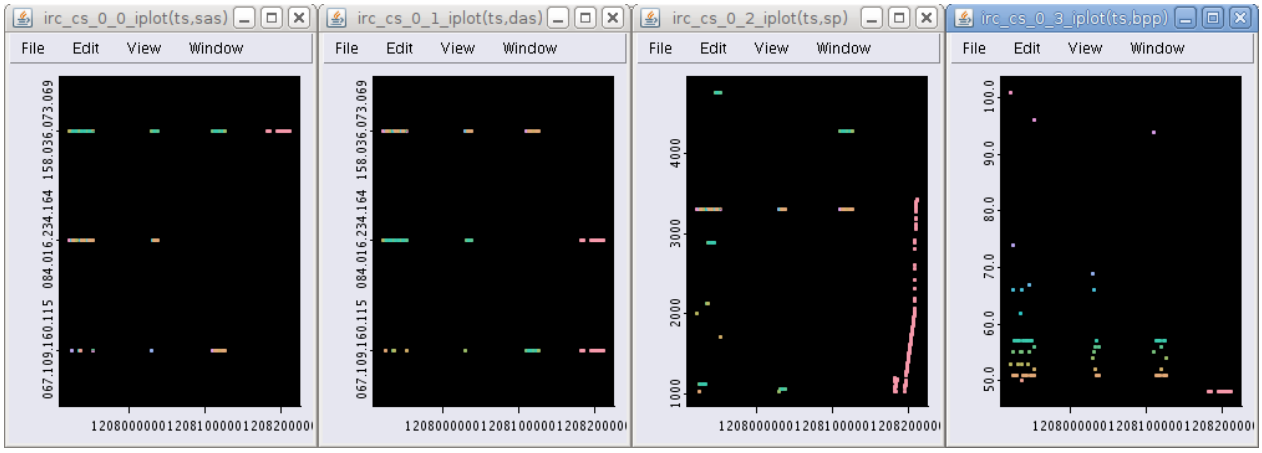
Figure 40 shows the plots brushed by the rainbow colors according to the sp attribute. Again the focus is on the connecting attempts from the bots after the isolation of the controllers. The beautiful rainbow spectrum in Figure 40 strongly suggests that the two controllers are of the same botnet because it is otherwise very rare that the bot can communicate with two controllers at the same time using two malwares of two different botnets.



(a) (b) (c) (d)

Figure 40: Scatter plots of data filtered by controllers and bots, color brushed by source port. Plots (a), (b), (c) and (d) are scatter plots of "sas vs ts", "das vs ts", "sp vs ts" and "bpp vs ts" respectively.

Figure 41 presents the plots brushed by the rainbow colors according to the bpp attribute. These plots simply shows that the payloads of the egress flows are quite similar to each other and have steady values, while the payloads of the ingress flows are much more various.

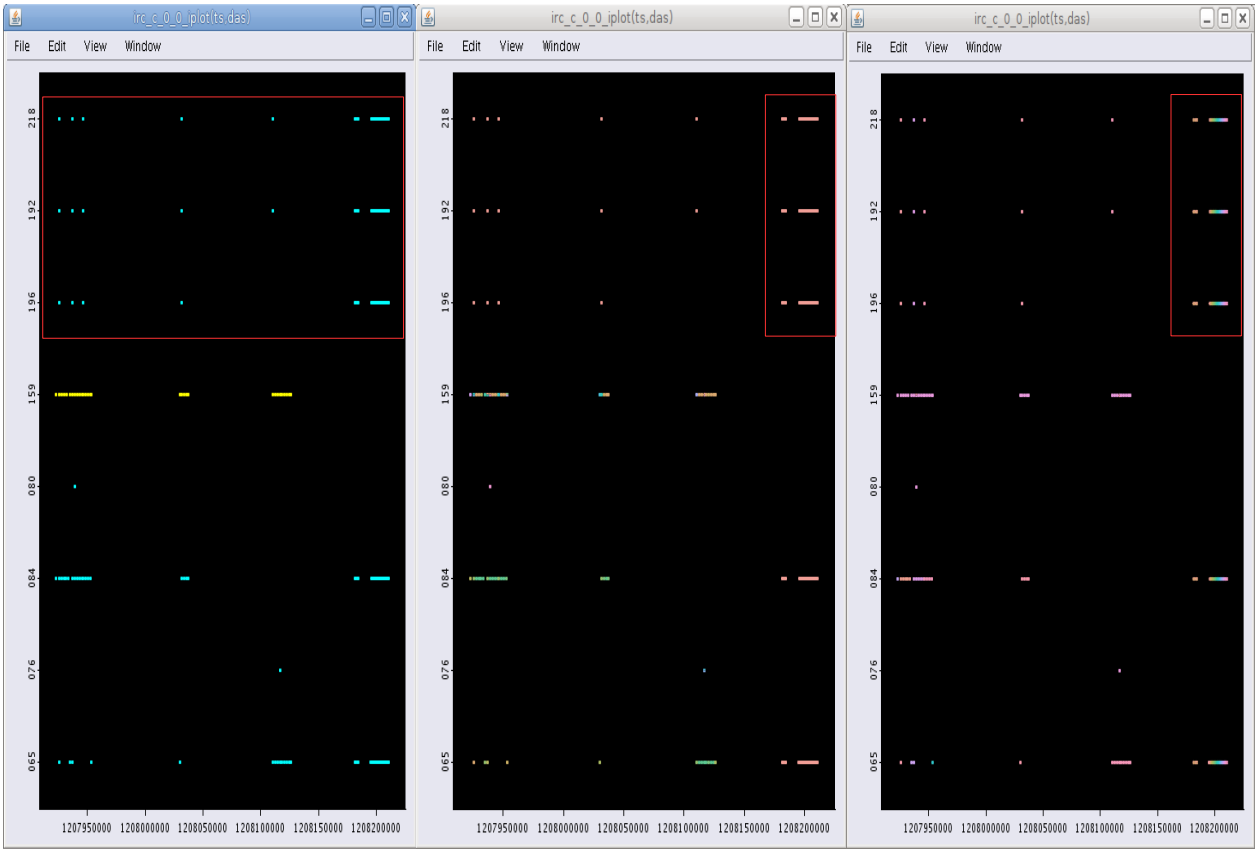


(a) (b) (c) (d)

Figure 41: Scatter plots of data filtered by controllers and bots, color brushed by byte per packet. Plots (a), (b), (c) and (d) are scatter plots of "sas vs ts", "das vs ts", "sp vs ts" and "bpp vs ts" respectively.

- Filter Bots in Order to Find Controllers

Because other scatter plots are similar to the ones presented in the first sub-dataset using the full filter expression, Figure 42 only shows the same scatter plots of "das vs ts", which are color brushed by direction, bpp and sp respectively. Figure 42 (a) shows the flows of three new possible controllers in the red box, which have very similar pattern especially the frequent connecting attempts after the known controllers are isolated. According to the "sas vs ts" scatter plot which is not presented here, none of these hosts respond to the bot. A hypothesis is that these newly appeared IP addresses are of the controllers within the same botnet. Hence these IP addresses are included in the malware planted in the bot. They did not respond probably because the IP addresses are not valid anymore or they are also isolated by the UNINETT CERT. The rainbow color spectra in the red boxes in Figure 42 (b) and (c) provide further evidences to support such hypothesis.

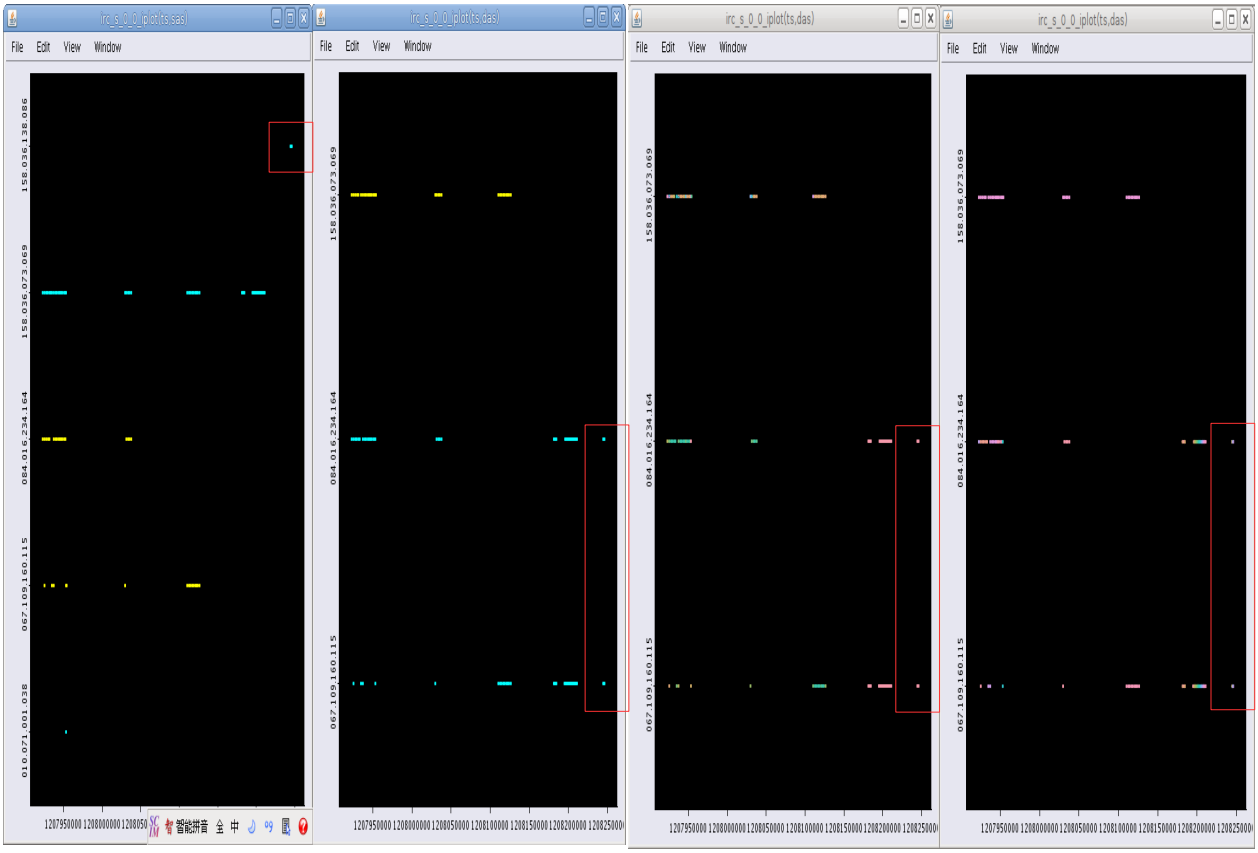


(a) (b) (c)

Figure 42: Scatter plots of "das vs ts", generated from data filtered by bots. Plots (a), (b), (c) are color brushed by flow direction, source port and byte per packet respectively.

- Filter Controllers in Order to Find Bots

Similar to Figure 42, Figure 43 showing in plots (b), (c) and (d) respectively the same scatter plots of "das vs ts", which are color brushed by direction, bpp and sp. In addition , Figure 43 (a) is the scatter plot of "sas vs ts". Figure 43 (a) and (b) show the flows of one new possible bot in the red box, which destined to contact both of the known controllers. Figure 43 (c) and (d) give more evidences to confirm that the patterns of the payload and of the source port are similar to the original C&C flows respectively.



(a) (b) (c) (d)

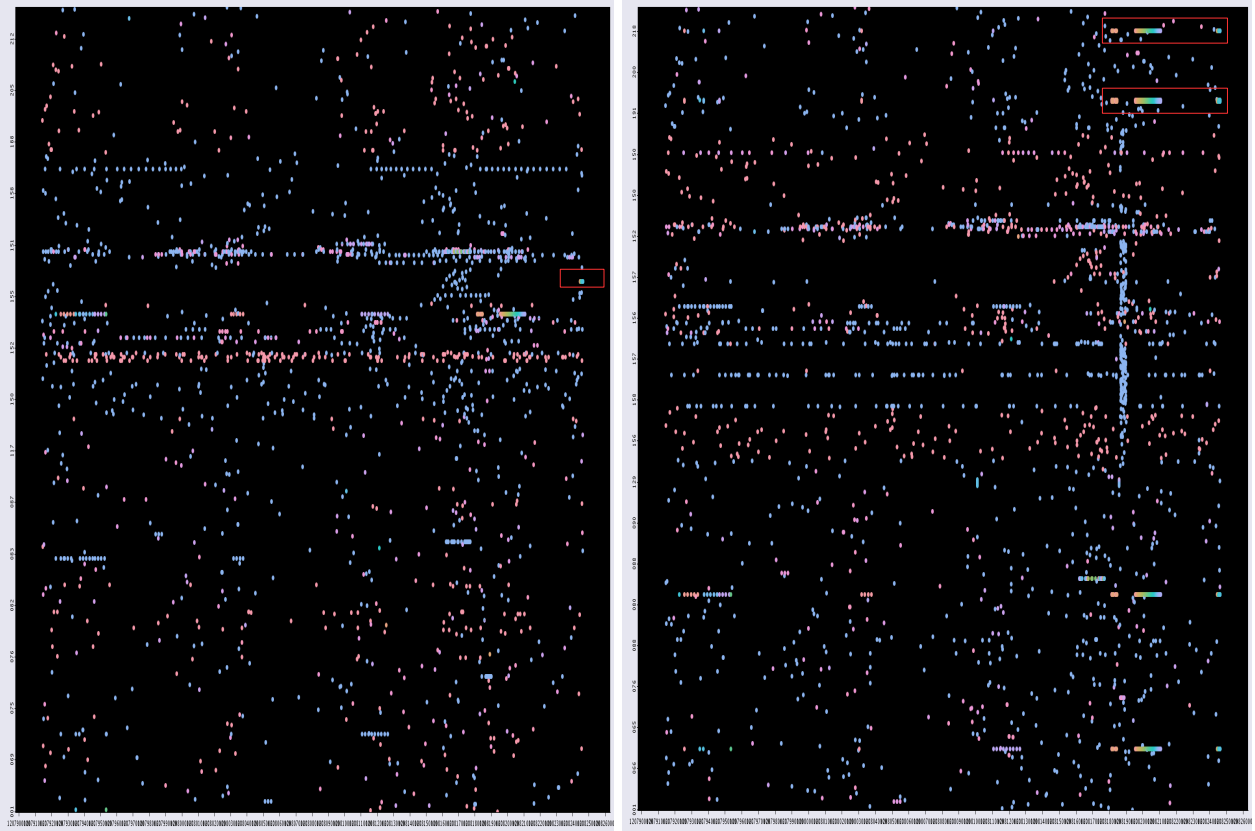
Figure 43: Scatter plots of data filtered by controllers. All plots are scatter plot of "das vs ts" exacept that plot (a) is scatter plot of "sas vs ts". Plots (a), (b), (c) and (d) are color brushed by flow direction, flow direction, source port and byte per packet respectively

- Filter Port Number in Order to Find Both Controllers and Bots

Although the visualizations of this sub-dataset have a lot of noise, the visual patterns observed from the visualization of the previous three sub-datasets are still clear. That is to say, periodical traffics from some hosts, and the rapid source port increase during the connecting attempts. The main results here are more evidences that further confirm the previous hypothesis.

Figure 44 presents the scatter plots of "sas vs ts" and "das vs ts" respectively in sub plots (a) and (b), which are both color brushed by sp. In the red box of Figure 44 (a) we identify the possible new bot discovered in Figure 43. In the red box of Figure 44 (b) we identify the possibly new controllers discovered in Figure 42 (further digging such as a zooming operation can show that there are three different controllers than two). Notice that there are more connecting attempts towards the suspicious new controllers then shown in Figure 42. As shown in Figure 43, such connecting attempts come from the suspicious new bot. Therefore, we can tell that the suspicious new bot and new controllers actually belong to the same botnet as the originally known bot and controllers do. Such result can not be concluded from any single one of the above four sub-

datasets.



(a) (b)

Figure 44: Scatter plots of data filtered by port number 3306. Plots (a), (b) are scatter plots of "sas vs ts" and "das vs ts" respectively. Both plots are color brushed by source port.

7.4.3 Application on Common Dataset

Unfortunately, because of the instinct low amount and sparse distribution of the traffics, the above patterns can not be identified anymore when applying the same investigation procedures as used above to the common dataset.

7.5 HTTP Botnet C&C Link

As mentioned in Section 7.1.1, the HTTP botnet dataset is used in the first part of this experimentation. Filtered by the known IP addresses of the controllers and bots, the dataset reduces most of the noise. Based on the discussions on the HTTP C&C links in Section 3.3, the main traffic phenomenon we look for here are also the initial malware downloading and the keep-alive signaling.

7.5.1 NetFlow Property Selection

The selection here is exactly as the one for the IRC botnet C&C link in Section 7.4.1.

7.5.2 Investigation

Global Overview Snapshot

As in the investigation for the IRC botnet C&C links, there are no immediate clear patterns in the global overview snapshot of the HTTP botnet dataset. Direct filtering by the know C&C links is applied in order to perform local drill-down to the traffics of the C&C links.

The IP address of the C&C controller is 64.*.*.*, which uses TCP port 80 to communicate with the infected bot 158.*.*.*. Also the same as in Section 7.4, in order to tell the direction of the flows in the plots when necessary, we denote the flows going from the controllers to the bots as ingress flows, and the flows of the other direction as egress flows.

Local Detail Drill-down

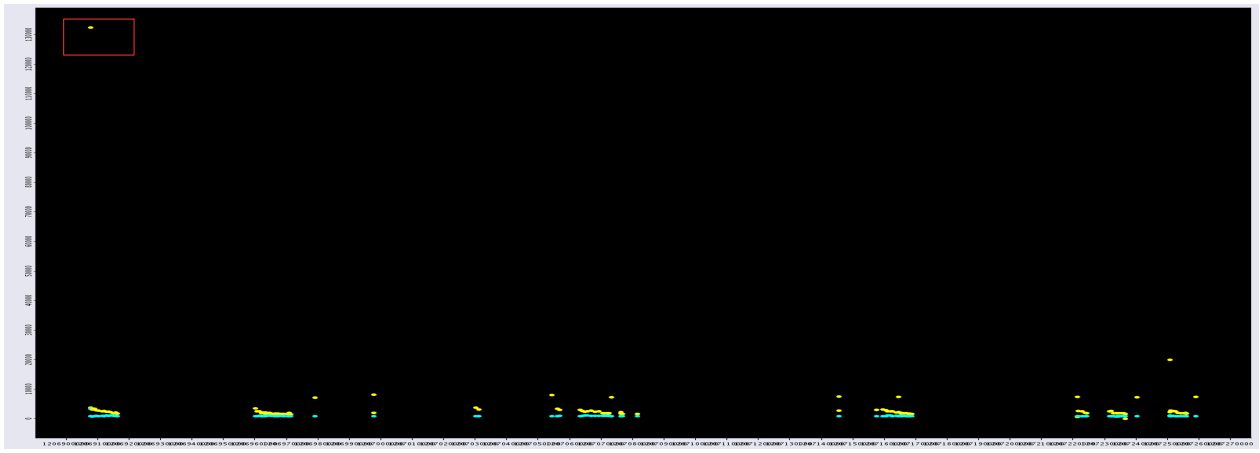
Basically, the techniques used in the actual experimentation of the HTTP botnet dataset such as the dataset sub-division, the plotting scheme and the color brushing schemes are exactly the same as in Section 7.4 for the IRC botnet dataset. Except that in this section, we mainly present the payload relevant visualizations. In other words, in each of the following sub-section, the three sub plots in each figure are scatter plots of “byt vs ts”, “pkt vs ts” and “bpp vs ts” respectively, which are color brushed according to the flow directions. Again, the egress flows are brushed as blue and the ingress flows are brushed as yellow.

- **Filter Controllers and Bots**

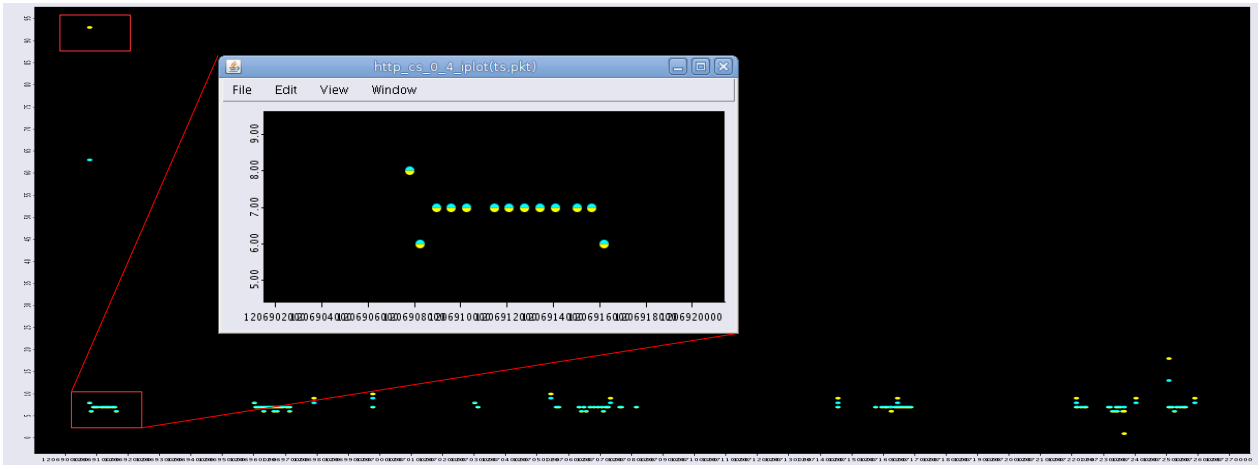
By a red box on the top-left conner of the plots in Figure 45, the data points indicating possible malware downloading activity are identified. Such data points are clearly distinguished from other data points by its extraordinary higher value. Additional packet analysis by the UNINETT CERT confirms that those flows are indeed malware downloading.

Apparently, most flows in each sub-figure of Figure 45 can be divided into two groups naturally by the color brushing according to the flow direction. In other words, the payload of the flows of each direction are different.

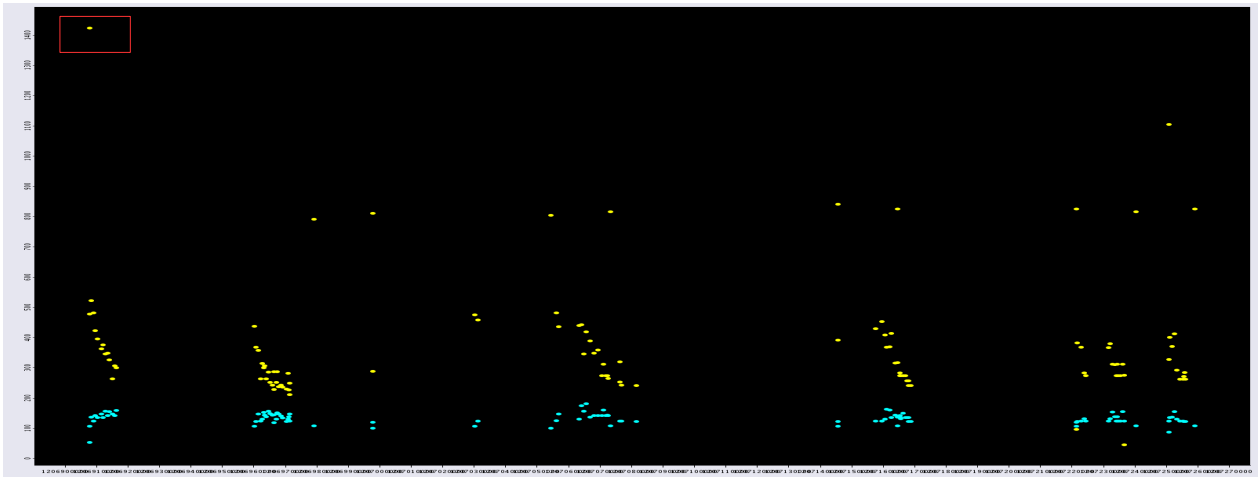
In addition, the values of the payload relevant attributes of each flow direction are quite stable. That is to say, it is very likely that the traffics of each flow direction is actually very similar. Though in Figure 45, it is hard to tell if the traffics are periodical, a zoom-in view within Figure 45 (b) shows that the traffics actually have a steady period of 10 minutes. This is an important evidence to support the hypothesis that the traffics are keep-alive signals of the HTTP botnet C&C link. Another evidence is that the downlink/uplink ratio seems too low for a normal HTTP traffics, because the HTTP servers usually transfer much more amount of data to their clients.



(a)



(b)

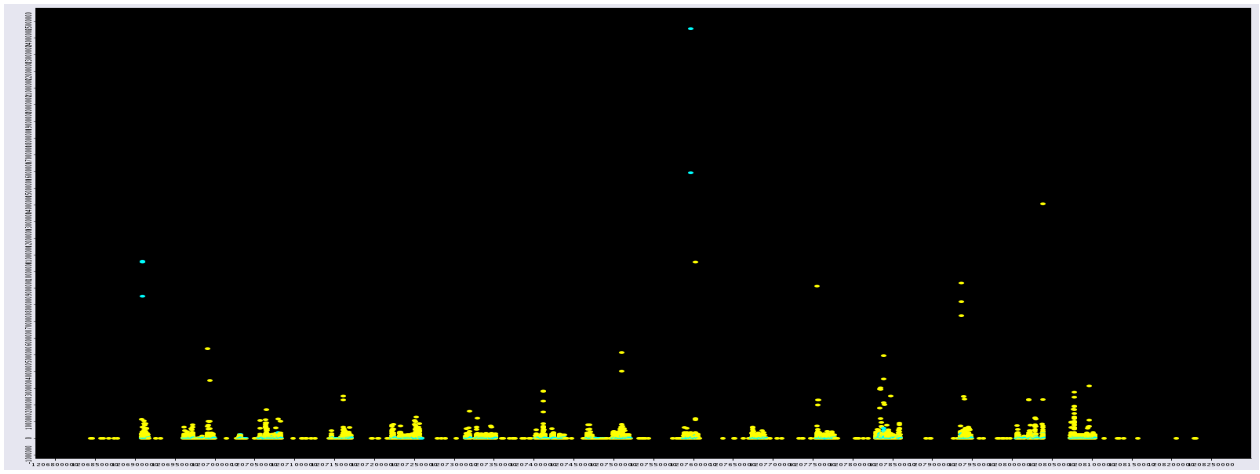


(c)

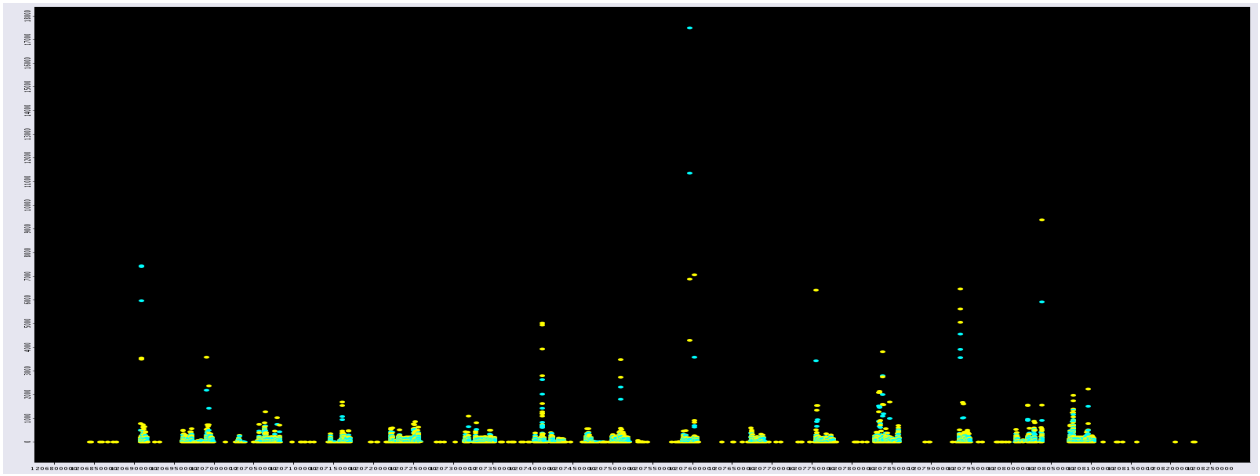
Figure 45: Scatter plots of data filtered by controllers and bots, color brushed by flow direction. Plots (a), (b) and (c) are scatter plots of "byt vs ts", "pkt vs ts" and "bpp vs ts" respectively.

- **Filter Bots in Order to Find Controllers**

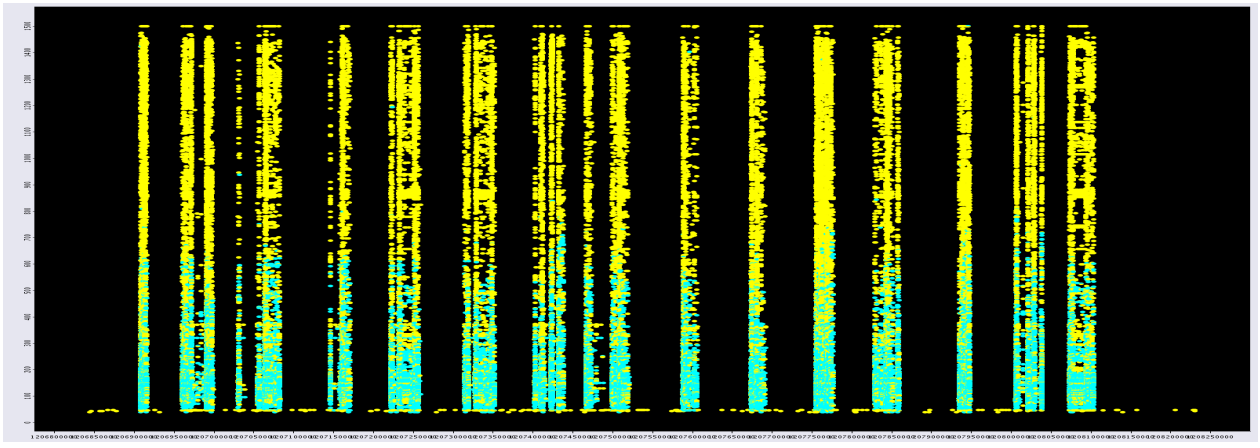
In the examination of this sub-dataset, we can not find new controllers because the noise is already too much. However, the large noise provides the information of the common HTTP traffics. Figure 46 shows that the amount of downlink traffics (brushed as yellow) are much higher than the amount of uplink traffics (brushed as blue).



(a)



(b)

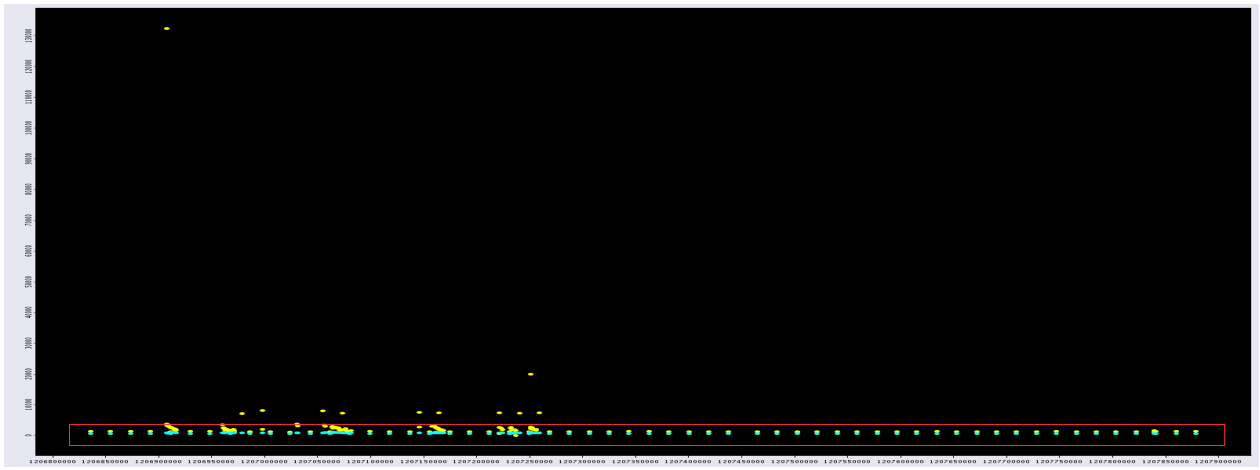


(c)

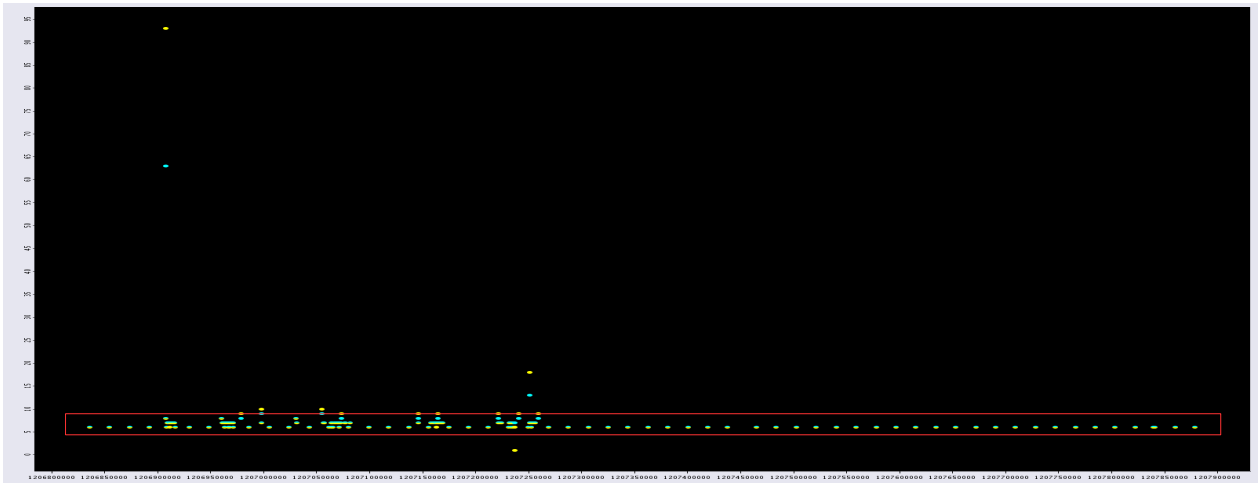
Figure 46: Scatter plots of data filtered by bots, color brushed by flow direction. Plots (a), (b) and (c) are scatter plots of "byt vs ts", "pkt vs ts" and "bpp vs ts" respectively.

- **Filter Controllers in Order to Find Bots**

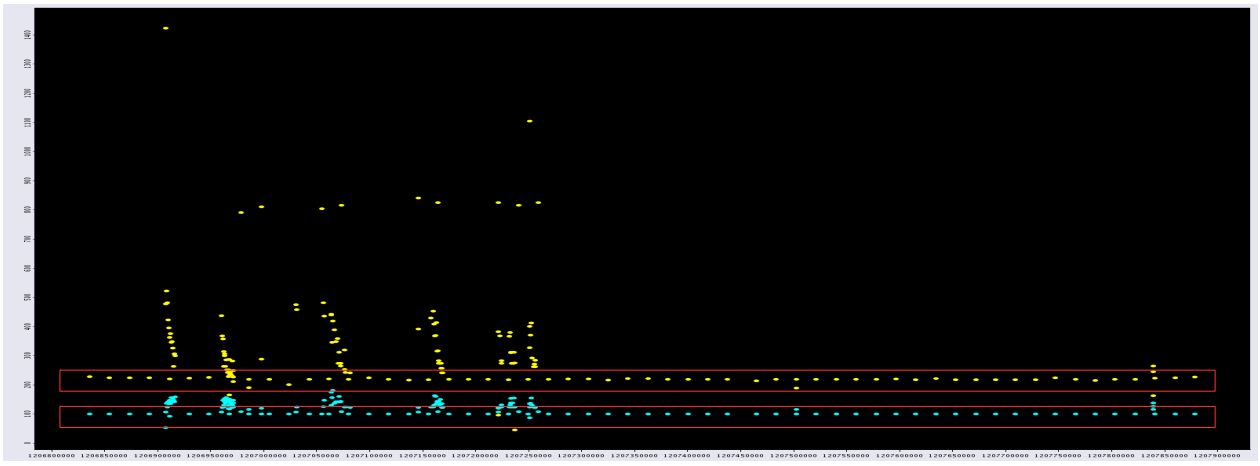
Luckily, we find one new suspicious bot connected to the known controller. As indicated in Figure 47, this bot have obvious periodical keep-alive traffics every 5 hours with the contorller. In addition, the amount of payload in the flows of both directions are also very stable, and the downlink/uplink ratio also seems quite low. They are both strong evidence suggesting the suspicious bot is actually an infected bot. However we can not find the data point representing the malware downloading.



(a)



(b)



(c)

Figure 47: Scatter plots of data filtered by controllers, color brushed by flow direction. Plots (a), (b) and (c) are scatter plots of "byt vs ts", "pkt vs ts" and "bpp vs ts" respectively.

- **Filter Port Number in Order to Find Both Controllers and Bots**

Because a great portion of the traffics in the Internet is HTTP traffics, this sub-dataset is simply too large for investigation by the UniVis.

7.5.3 Application on Common Dataset

Unfortunately again, just as in the experimentation for the IRC botnet dataset, none of the above patterns can be identified anymore when applying the same investigation procedures to the common dataset.

8 Discussion

In Chapter 7, the experimentation of the identification of three kinds of malicious activities are described. In this chapter, the experimentation is discussed against the practical criteria raised in Section 1.3. Briefly, the actual detection performance and the way to improve it in practice are of the utmost concern.

8.1 Detection of Scanning

Basically, the detection of the scanning in our experiments is effective and accurate, even when it is applied to the sampled common dataset, except for the IP-sweep scanning with very limited packet number in each flow.

Compared to the traditional statistical methods, where only the top talkers with the highest traffic statistics such as the throughput or the number of flow are identified, the visual methods in the UniVis can identify scanning activity without prominent traffic statistics as well. More importantly, the identification of multiple scanning activities can be done very quickly at a glance of a single global overview snapshot produced by the UniVis, whereas by using the statistical methods the same work is carried out much more slowly by reading through the listed top talkers line by line.

8.2 Detection of IRC Botnet C&C Link

For the detection of the IRC botnet C&C links, the main visual patterns we found in our experiments are periodical data points that form lines, and the colorful rainbow spectrum caused by rapid increase of the source port when a bot is trying to connect to a controller who suddenly does not respond any more.

In addition, combining the investigation results of several subsets derived from the same dataset can actually leads to new conclusions or help consolidate previous hypothesis.

However, without the initial filtering process to narrow down the target dataset, it is very unlikely to discover the above patterns among massive noises, except that the periodical flows are enhanced by suitable mathematical algorithms such as correlation analysis. Traditional top talker statistics provided directly by the nfdump are less useful in this case.

8.3 Detection of HTTP Botnet C&C Link

As for the detection of the HTTP botnet C&C links, we observed three visual patterns. First, a peak value of byte per flow indicates the malware downloading activity. Second, the amount of flow payloads are stable and different for each direction of flow, meanwhile the downlink/uplink ratio is low. Third, the traffics are usually periodical.

In terms of the identification of such patterns, the first pattern can easily disappear in slightly worse noise condition. Considering the patterns of stable the low downlink/uplink ratio and of

periodical keep-alive signal, considerable amount of legitimate websites also have the same characteristic. Therefore, more efforts are still required to distinguish such benign sites from the malicious ones.

8.4 Essential Factors for Improvement of Detection

According to the experimentation and the above discussions on it, in order to improve the detection results, the following three factors are essential.

First, the VizSEC system used as the detection tool must scale very well. Such scalability includes the capacity to take in enormous amount of data, and fast enough processing speed for the preprocessing and visualization of huge data, and real-time interactions with the visualizations.

Second, deeper understanding of target malicious activities should be the preliminary knowledge acquired by the users of the VizSEC system. Such knowledge includes the theoretical features of the malicious activities, and more importantly, the practical experiences of the visual exploration of such activities by using the VizSEC system.

Last but not the least, the VizSEC system should well address the usability issues with regards to human perception capabilities and limitations.

8.5 Ideal Working Scenario

Taking an ISP CERT such as the UNINETT CERT for example, an ideal work scenario of network security management routine by using a VizSEC system can be as follows. When the new input data arrive, the VizSEC system automatically pre-processes and produces the overview snapshot of the input data, and finally logs the visualization results according to a predefined schedule. The human administrator check the overview snapshots, which should not occupy much effort, and then dig into more details of the suspicious spots. Because the whole procedure of the drill-down investigation can be logged as well, human administrators can think over the case independently, then discuss with their colleagues over the logs, and finally add the new procedures in the automated tasks scheduled for the VizSEC system.

9 Conclusion and Future Work

9.1 Conclusion

To sum up, in this thesis work we first survey on the existing VizSEC systems. Then, according to the survey results, we design and implement a VizSEC system named UniVis, which is built mainly by integrating the existing utilities such as the nfdump, the R and the iPlots together, also with special features such as the visualization tree. The UniVis distinguishes itself from other VizSEC systems by the concentration on the NetFlow data, the utilization of interactive visualization, and the design taking human factors into account at the very beginning.

Second, using the UniVis, real network traffics are used for the experimentation of detecting several typical malicious network activities. The experiment results can be used directly to detect such malicious activities, or to narrow down the range of the suspicious dataset.

Finally, practical challenges are pointed out and the corresponding improvement suggestions are given, based on the practical experiences gained from the above work of survey, design, implementation and experimentation.

9.2 Future work

According to the discussion of the essential factors for the improvement of the detection results in Section 8.4, the future work should focus on those factors.

For the first factor, the scalability of UniVis can be improved by replacing the current Java based visualization modules with more efficient implementation. For instance by moving the graphic computing from the CPU to the graphic processing unit (GPU) can give a promising boost to the processing speed.

For the second factor, more research on malicious network activities should be done. No doubt this endeavor will be tough and endless. The VizSEC systems such as our UniVis can surely be used for such work as an complementary approach to the others.

The third factor on one hand depends on the development of the human perception theory, which is not quite the task of the security domain. On the other hand, it depends on better understanding of user requirements and experiences, which can be improved by carefully designed user study of the users of the UniVis system.

References

1: Cisco NetFlows, http://www.cisco.com/web/go/netflow/
2: TCPDump/LIBPCAP, http://www.tcpdump.org/
3: Russ McRee, Security Visualization: What you don't see can hurt you, 2008
4: Edward R. Tufte, The visual display of quantitative information, 1986
5: Colin Ware, Information visualization: perception for design, 2000
6: Edward R. Tufte, Visual Explanation, 1997
7: VizSEC Workshop, http://www.vizsec.org/
8: NfSen, http://nfsen.sourceforge.net/
9: MRTG, http://oss.oetiker.ch/mrtg/
10: Gregory Conti and Kulsoom Abdullah and Julian Grizzard and John Stasko and John A. Copeland and Mustaque Ahamad and Henry L. Owen and Chris Lee, Countering Security Information Overload through Alert and Packet Visualization, 2006
11: Kulsoom Abdullah and Chris Lee and Gregory Conti and John A. Copeland and John Stasko, IDS RainStorm: Visualizing IDS Alarms, 2005
12: Snort, http://www.snort.org/
13: Gregory Conti and Julian Grizzard and Mustaque Ahamad and Henry Owen, Visual Exploration of Malicious Network Objects Using Semantic Zoom, Interactive Encoding and Dynamic Queries, 2005
14: William Yurcik, Visualizing NetFlows for security at line speed: the SIFT tool suite, 2005
15: Two visual computer network security monitoring tools incorporating operator interface requirements, citeseer.ist.psu.edu/yurcik03two.html
16: A Prototype Tool for Visual Data Mining of Network Traffic for Intrusion Detection, citeseer.ist.psu.edu/yurcik03prototype.html
17: Kiran Lakkaraju and William Yurcik and Adam J. Lee, NVisionIP: netflow visualizations of system state for security situational awareness, 2004
18: Xiaoxin Yin and William Yurcik and Adam Slagell, The Design of VisFlowConnect-IP: A Link Analysis System for IP Security Situational Awareness, 2005
19: Xiaoxin Yin and William Yurcik and Michael Treaster and Yifan Li and Kiran Lakkaraju, VisFlowConnect: netflow visualizations of link relationships for security situational awareness, 2004
20: William Yurcik, Tool update: visflowconnect-IP with advanced filtering from usability testing, 2006
21: Argus NetFlows, http://www.qosient.com/argus/
22: Anita D. D'Amico and John R. Goodall and Daniel R. Tesone and Jason K. Kopylec, Visual Discovery in Computer Network Defense, 2007
23: Inxight Software, http://www.inxight.com/
24: Pin Ren and Yan Gao and Zhichun Li and Yan Chen and Benjamin Watson, IDGraphs: Intrusion Detection and Analysis Using Histographs, 2005
25: Jean-Pierre van Riel and Barry Irwin, InetVis, a visual tool for network telescope traffic analysis, 2006

26: Stephen Lau, The Spinning Cube of Potential Doom, 2004
27: AfterGlow, http://afterglow.sourceforge.net/
28: GraphViz, http://www.graphviz.org/
29: What a Botnet Looks Like,
http://www.csoonline.com/article/348317/What_a_Botnet_Looks_Like
30: Florian Mansmann and Daniel A. Keim and Stephen C. North and Brian Rexroad and Daniel Sheleheda, Visual Analysis of Network Traffic for Resource Planning, Interactive Monitoring, and Interpretation of Security Threats, 2007
31: UNINETT, http://www.uninett.no/
32: Ben Shneiderman, The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations, 1996
33: Ben Shneiderman, Inventing discovery tools: combining information visualization with data mining, 2002
34: John R. Goodall and Wayne G. Lutters and Anita Komlodi, I know my network: collaboration and expertise in intrusion detection, 2004
35: Gregory Conti and Mustaque Ahamad and John Stasko, Attacking information visualization system usability overloading and deceiving the human, 2005
36: Gregory John Conti, Countering network-level denial of information attacks using information visualization, 2006
37: Patrick Edward Hoffman, Table visualizations: a formal model and its applications, 2000
38: ARBOR NETWORKS, WORLDWIDE INFRASTRUCTURE SECURITY REPORT, 2007
39: RFC4732: Internet Denial-of-Service Considerations, http://tools.ietf.org/html/rfc4732
40: Kiran Lakkaraju and Ratna Bearavolu and Adam Slagell and William Yurcik and Stephen North, Closing-the-Loop in NVisionIP: Integrating Discovery and Search in Security Visualizations, 2005
41: NfDump, http://nfdump.sourceforge.net/
42: R, http://www.r-project.org/
43: iPlots, http://www.rosuda.org/iplots/
44: CRAN, http://cran.r-project.org/
45: JGR, http://jgr.markushelbig.org/JGR.html
46: Morten Knutsen, Fighting Botnets in an Internet Service Provider Environment, 2005

Appendix A Modification on Nfdump

A.1 Additions in nf_common.c

```
> static const char *
> inet_ntop_v4(const void *src, char *dst, size_t size)
> {
>     const char digits[] = "0123456789";
>     int i;
>     struct in_addr *addr = (struct in_addr *)src;
>     u_long a = ntohl(addr->s_addr);
>     const char *orig_dst = dst;
>
>     if (size < INET_ADDRSTRLEN) {
>         errno = ENOSPC;
>         return NULL;
>     }
>     for (i = 0; i < 4; ++i) {
>         int n = (a >> (24 - i * 8)) & 0xFF;
>         int non_zerop = 0;
>
>         if (non_zerop || n / 100 > 0) {
>             *dst++ = digits[n / 100];
>             n %= 100;
>             non_zerop = 1;
>         }else *dst++ = '0';
>         if (non_zerop || n / 10 > 0) {
>             *dst++ = digits[n / 10];
>             n %= 10;
>             non_zerop = 1;
>         }else *dst++ = '0';
```

```
>         *dst++ = digits[n];
>         if (i != 3)
>             *dst++ = '.';
>     }
>     *dst++ = '\0';
>     return orig_dst;
> }
>
> static void Vis_String_SrcAddr(master_record_t *r, char *string) {
> char tmp_str[IP_STRING_LEN];
>
>         tmp_str[0] = 0;
>         uint32_t          ip;
>         ip = do_anonymize ? anonymize(r->v4.srcaddr) : r->v4.srcaddr;
>         ip = htonl(ip);
>         inet_ntop_v4(&ip, tmp_str, sizeof(tmp_str));
>
>         tmp_str[IP_STRING_LEN-1] = 0;
>
>         snprintf(string, MAX_STRING_LENGTH-1, "%15s", tmp_str);
>
>         string[MAX_STRING_LENGTH-1] = 0;
>
>
> } // End of Vis_String_SrcAddr
>
> static void Vis_String_DstAddr(master_record_t *r, char *string) {
> char tmp_str[IP_STRING_LEN];
>
>         tmp_str[0] = 0;
>         uint32_t          ip;
```

```
>       ip = do_anonymize ? anonymize(r->v4.dstaddr) : r->v4.dstaddr;
>       ip = htonl(ip);
>       inet_ntop_v4(&ip, tmp_str, sizeof(tmp_str));
>
>       tmp_str[IP_STRING_LEN-1] = 0;
>
>       snprintf(string, MAX_STRING_LENGTH-1, "%15s", tmp_str);
>
>       string[MAX_STRING_LENGTH-1] = 0;
>
>
> } // End of Vis_String_DstAddr
>
> void flow_record_to_vismix(void *record, uint64_t numflows, char ** s, int anon, int tag) {
> uint32_t      sa, da;
> master_record_t *r = (master_record_t *)record;
> char          first_str[64], last_str[64], prot_str[6], sa_str[IP_STRING_LEN],
da_str[IP_STRING_LEN], flags_str[16];
>
>       // Assume all flows are IPv4
>       if ( anon ) {
>               r->v4.srcaddr = anonymize(r->v4.srcaddr);
>               r->v4.dstaddr = anonymize(r->v4.dstaddr);
>       }
>
>       // Make sure Endian does not screw us up
>    sa = r->v6.srcaddr[1] & 0xffffffffLL;
>    da = r->v6.dstaddr[1] & 0xffffffffLL;
>
>       String_FirstSeen(r, first_str);
>       String_LastSeen(r, last_str);
>       String_Protocol(r, prot_str);
```

```
>       Vis_String_SrcAddr(r, sa_str);
>       Vis_String_DstAddr(r, da_str);
>       String_Flags(r,flags_str);
>
>       snprintf(data_string, STRINGSIZE-1 ,"%u,%s,%u,%s,%u,%u,%s,%u,%s,%u,%u,%s,%u,
%u,%s,%llu,%llu",
>                               r->first, first_str, r->last, last_str, r->last - r->first, r->prot, prot_str,
>                               sa, sa_str, r->srcport, da, da_str, r->dstport,
>                               r->tcp_flags, flags_str, (unsigned long long)r->dPkts, (unsigned
long long)r->dOctets);
>
>       data_string[STRINGSIZE-1] = 0;
>
>       *s = data_string;
>
> } // End of flow_record_vismix
>
```

Appendix B Shell Scripts

B.1 dump.sh

```
#!/bin/bash
# dump binary nfdump data to readable csv data with customized flow records.
# $1 the output (customized csv data) file name.
# $[2-*] the option for nfdump

# output path
out=$1

# copy header to output file
cat scripts/vismix.hdr > $out

# shift the arguments left so the new argument array are all for nfdump
shift 1

# using modified nfdump to dump in customized format
nfdump -q -o vismix $* >> $out
```

B.2 preload.sh

```
#!/bin/bash
# convert .csv file to R workspace .ws file.
# $1 the output path.
# $2 the data name in workspace.

# output path
out=$1

if [ ! -f $out.csv ]
```

```
then
  echo "Input file [$out.csv] not found - Aborting"
    exit -1
fi

# data name in workspace
name=$2

# temporary R script name
script=$out'_preload.r'

# create temporary R script
#cat scripts/r_vismix.hdr > $script
echo $name' <- read.table("'$out.csv'",header=T,sep=",", quote="");' >> $script
echo 'save.image("'$out.ws'",compress=TRUE)' >> $script

# execute R script in BATCH mode
R --no-save --slave < $script

# remove temporary R script
rm $script

# remove csv file
#rm $out.csv
```

B.3 getflownumber.sh

```
#!/bin/bash
# 1st parameter is raw binary nfdump file name
# find the line records overall flow number and get the flow number
nfdump -r $1 -I | grep 'Flows:' | sed 's/Flows: //g'
```

B.4 gettimewindow.sh

```
#!/bin/bash
# find the line records first/last seen time in UNIX timestamp format
# $* the option for nfdump

firstts=$(nfdump -I $* | grep 'First:' | sed 's/First: //g')
lastts=$(nfdump -I $* | grep 'Last:' | sed 's/Last: //g')

if [ -z "$firstts" ] || [ -z $lastts ]
then
        exit 1
fi

# the following commands do the same thing: convert UNIX timestamp format to customized format
#date -d @$first +%Y/%m/%d.%T

first=$(date -d '1970-01-01 '$firstts' sec GMT' +%Y/%m/%d.%T)
last=$(date -d '1970-01-01 '$lastts' sec GMT' +%Y/%m/%d.%T)
echo $first-$last
```

Appendix C R Scripts

The following use dataset “irc_cs.ws” as example.

C.1 Load

load("/home/jin/data/workspace/final/irc_cs.ws")

C.2 Scatter Plot

0> iset.set('irc_cs_0')

1> attach(iset("irc_cs_0")[])

2> iplot(ts,sas, customFieldBg=TRUE, COL_CUSTOMBG="black", ptDiam=3)

3> detach(iset("irc_cs_0")[])

4> .jcall(.jcall(.iplots[[as.integer(iplot.cur())]]
$obj,"Ljava/awt/Frame;","getFrame"),"V","setState",as.integer(1))

5> .jcall(.jcall(.iplots[[as.integer(iplot.cur())]]
$obj,"Ljava/awt/Frame;","getFrame"),"V","setVisible",FALSE)

6> iplot.size(1,1)

7> iplot.opt(title="irc_cs_0_0_iplot(ts,sas)")

8> iplot.opt(fillColor="green")

C.3 Parallel Coordinate Plot

0> iset.set('irc_cs_0')

1> attach(iset("irc_cs_0")[])

2> iplot(ts,sas, customFieldBg=TRUE, COL_CUSTOMBG="black", ptDiam=3)

3> detach(iset("irc_cs_0")[])

4> .jcall(.jcall(.iplots[[as.integer(iplot.cur())]]
$obj,"Ljava/awt/Frame;","getFrame"),"V","setState",as.integer(1))

5> .jcall(.jcall(.iplots[[as.integer(iplot.cur())]]
$obj,"Ljava/awt/Frame;","getFrame"),"V","setVisible",FALSE)

6> iplot.size(1,1)

7> iplot.opt(title="irc_cs_0_0_iplot(ts,sas)")

8> iplot.opt(fillColor="green")

C.4 Histogram Plot

```
0> iset.set('irc_cs_0')

1> attach(iset("irc_cs_0")[])

2> ihist(ts)

3> detach(iset("irc_cs_0")[])

4> .jcall(.jcall(.iplots[[as.integer(iplot.cur())]]
$obj,"Ljava/awt/Frame;","getFrame"),"V","setState",as.integer(1))

5> .jcall(.jcall(.iplots[[as.integer(iplot.cur())]]
$obj,"Ljava/awt/Frame;","getFrame"),"V","setVisible",FALSE)

6> iplot.size(1,1)

7> iplot.opt(title="irc_cs_0_0_ihist(ts)")

8> iplot.opt(borderColor="black")
```

C.5 Bar Plot

```
0> iset.set('irc_cs_0')

1> attach(iset("irc_cs_0")[])

2> ibar(ts)

3> detach(iset("irc_cs_0")[])

4> .jcall(.jcall(.iplots[[as.integer(iplot.cur())]]
$obj,"Ljava/awt/Frame;","getFrame"),"V","setState",as.integer(1))

5> .jcall(.jcall(.iplots[[as.integer(iplot.cur())]]
$obj,"Ljava/awt/Frame;","getFrame"),"V","setVisible",FALSE)

6> iplot.size(1,1)

7> iplot.opt(title="irc_cs_0_1_ibar(ts)")

8> iplot.opt(borderColor="black")
```

C.6 Filter Color Brushing

```
0> iset.set("irc_cs_0")

1> color <- rep(3, dim(iset(iset.cur()))[1])

2> data <- irc_cs

3> attach(data)

4> color[dp==25] <- 05
```

```
5> color[sp==25] <- 07
6> detach(data)
7> rm(data)
8> iplot.opt(col=color)
9> rm(color)
```

C.7 Rainbow Color Brushing

```
0> source("scripts/rainbowcolor.r")
1> iset.set("irc_cs_0")
2> iset.col(rainbowcolor(iset("irc_cs_0")[]$ts))
3> rm(rainbowcolor)
```

C.8 rainbowcolor.r

```
# Get the brush color for the input data
# by 64 rainbow color space evenly according to the value of the data
rainbowcolor <- function(x) {
  lx <- levels(factor(x))
  len <- length(lx);
  palette <- rep(64, len)
  color <- rep(1, length(x))
  if (len > 64){
    delta <- ifelse(len %% 64 > 0, len %/% 64 +1, len %/% 64)
    remainder <- len %% delta
    count <- len %/% delta
    for (i in 0:(count - 1)) {
      palette[(i * delta + 1)  : ((i + 1) * delta)] <- rep(64 + i, delta)
    }
    if (remainder > 0){
      palette[(count * delta + 1) : len] <- rep(64 + count, len - (count * delta))
    }
  }else if (len <= 64) {
```

```
    delta <- 64 / len
    for (i in 0:(len - 1)) {
      palette[i + 1] <- 64 + round(delta * i)
    }
  }

  for (i in 1:len){
    color[x==lx[i]] <- palette[i]
  }

  return (color)
}
```

C.9 Save Plot

```
0> iset.set("irc_cs_0")
1> iplot.set(2)
2> p<-.iplots[[as.integer(iplot.cur())]]
3> com=.jcall(p$obj,"Ljava/awt/Component;","getComponent")
4> h<-.jcall(com,"I","getHeight")
5> w<-.jcall(com,"I","getWidth")
6> .jcall(com,,"setSize", as.integer(1600), as.integer(1200))
7> img=.jnew("java/awt/image/BufferedImage",as.integer(1600),as.integer(1200),as.integer(1))
8> g=.jcall(img,"Ljava/awt/Graphics;","getGraphics")
9> .jcall(com,,"paint",g)
10>
.jcall("javax/imageio/ImageIO","Z","write",.jcast(img,"java/awt/image/RenderedImage"),"png",.jnew("java/io/File", "/home/jin/data/workspace/irc_cs_0_1_ibar(ts).png"))
11> .jcall(com,,"setSize", as.integer(w), as.integer(h))
12> rm(p)
13> rm(h)
14> rm(w)
15> rm(com)
```

```
16> rm(img)
17> rm(g)
```