User-centred security event visualisation

Christopher Humphries

ANNÉE 2015

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique
Ecole doctorale Matisse

présentée par

Christopher Humphries

préparée à l'unité de recherche CIDre
Confidentialité, Intégrité, Disponibilité et Répartition
Supélec / INRIA

Visualisation d'événements de sécurité centrée autour de l'utilisateur

---

User-centred security event visualisation

Thèse soutenue à Rennes
le 8 décembre 2015
devant le jury composé de :

Hervé Debar
Télécom SudParis / rapporteur

Jean-Marc Robert
École de Technologie Supérieure / rapporteur

Isabelle Chrisment
Télécom Nancy / examinatrice

Christophe BIDAN
CentraleSupélec / directeur de thèse

Nicolas PRIGENT
CentraleSupélec / co-directeur de thèse

Frédéric Majorczyk
DGA-MI / co-directeur de thèse

I didn’t go to university. Didn’t even finish A-levels. But I have sympathy for those who did.

— Terry Pratchett

RÉSUMÉ EN FRANÇAIS

Alors que les systèmes d'information deviennent de plus en plus complexes, il n'est plus possible d'en maitriser tous les aspects et les mécanismes de sécurité traditionnels atteignent leurs limites. Les systèmes de détection d'intrusion ont été proposés pour faire face à ce nouveau défi. Ils offrent une approche automatique et rapide pour détecter et répondre aux intrusions en identifiant des motifs connus ou des situations qui divergent du comportement normal. Ils sont excellents pour répéter cette action. Cependant, de tels systèmes réagissent mal aux événements inconnus, déclenchent beaucoup de faux positifs et manquent des événements importants. Ainsi, il est aujourd'hui de plus en plus difficile de gérer les énormes quantités de données produites dans le cadre de la sécurité des systèmes d'information.

Pour cette raison, les opérateurs humains sont toujours nécessaires pour donner du sens aux événements de sécurité rapportés. Pour que leurs interventions soient utiles (en particulier quand la tâche est d'analyser des alertes), ces opérateurs ont besoin de comprendre des situations rapidement, d'obtenir facilement une vision globale et les réponses à des questions, ainsi que de consulter des grandes quantités de données contextuelles annexes.

La recherche en analyse des évènements de sécurité s'est récemment portée sur la visualisation. La visualisation pour la sécurité se place entre des solutions manuelles et automatiques, et vise à combiner le meilleur des deux pour proposer des outils efficaces en pratiques. En d'autres termes, des représentations visuelles permettent d'améliorer le processus de supervision et de fouille de données du point de vue de l'opérateur en améliorant la manière dont les données de sécurité lui sont communiquées.

La conception d'outils de visualisation dédiés à la sécurité est rendue difficile par plusieurs aspects : les formats de données et les protocoles sont variés, les standards sont nombreux et le matériel et les logiciels informatiques sont souvent paramétrés de manière très fine par les administrateurs pour répondre à leurs besoin spécifiques. Alors que certains domaines peuvent compter sur la nature immuable des problèmes ou la disponibilité de temps, les problèmes en sécurité sont trop souvent nouveaux et exigent en outre une attention immédiate.

⋈

Le processus de visualisation peut être vu comme une série de transformations de données. Chaque étape peut être spécifiée en utilisant une grammaire graphique, ce qui permet de construire une description concrète pour chaque visualisation.

À son niveau le plus élémentaire, une visualisation est un assemblage de composants graphiques, chacun ayant des attributs paramétrable. Ces variables visuelles peuvent être associées aux données pour obtenir une transcription graphique d'un sous-ensemble de ces données. La perception humaine étant hétérogène et changeante, chaque variable visuelle est plus ou moins adaptée pour transmettre les informations à l'utilisateur. Chacune d'entre elles doit donc être soigneusement sélectionnée en fonction des objectifs de la visualisation.

Lorsqu'on prend en considération l'expérience utilisateur, un nouveau processus d'interaction homme-machine émerge. Des boucles dans les étapes permettent des retours des utilisateurs et représentent un processus de fouille adopté pendant l'exploration de données. Pendant toutes ces interactions, les données graphiques doivent être retraduites mentalement par l'utilisateur. Pour alléger la charge cognitive induite, la progression de la fouille doit guider l'utilisateur et suivre une progression logique narrative proche de sa façon de penser.

⋈

Actuellement, la visualisation d'événements de sécurité suit une approche et des objectifs adoptés par des équipes chargés de la sécurité réactive des systèmes, et vise au moins un but parmi trois :

monitoring Dans le cadre de la surveillance des systèmes et des réseaux, les opérateurs utilisent des tableaux de bord de visualisations pour s'assurer que les métriques sont dans des plages optimales, dans le but de garantir la disponibilité des services et chercher des signes d'attaques connues et de comportements malveillants.

analyse Lorsque des anomalies ou des intrusions sont signalées par des systèmes de détection ou des opérateurs avec des outils de monitoring, des outils d'analyse visuelle sont utilisés pour mieux comprendre ces anomalies/intrusions. En explorant les données de sécurité, les analystes cherchent à trouver des explications pour ces incidents, à recréer les scénarios des attaques et à caractériser les faux négatifs.

reporting Une fois qu'assez d'informations ont été recueillis, elles ont souvent besoin d'être transmises au reste de l'équipe ou à des entités externes. Dans ces cas les rapports ont pour objectif d'informer des collègues, des cadres ou des externes dans le cas ou la gestion de la crise auprès des médias par exemple.

Pour accomplir chacun de ces objectifs, les outils vont proposer des visualisations adaptées. La surveillance favorisera des visualisations adaptées à la compréhension des données en temps réel et compatible à les données en évolution, donc plus simples et facile à appréhender. Puisque l'exploration de données est moins sujet aux contraintes du temps réel, les visualisation peuvent être plus complexes et configurable par l'utilisateur. Les rapports communiquant les résultats des deux étapes précédentes doivent condenser toutes les informations pour expliquer la situation ainsi qu'un contexte assez riche pour palier le manque d'interactivité.

⋈

La construction des outils de visualisation nécessite les connaissances de multiples domaines tels que les statistiques, la conception d'interfaces et la psychologie. Les experts en sécurité manquent souvent d'expérience dans ces domaines ce qui rend la fabrication d'outils ad hoc de visualisation difficile. Ils sont avant tout experts en sécurité, mais rarement entrainé pour la visualisation. Même si la visualisation sans entrainement fais parfois preuve d'innovation, elle peut aussi produire des résultats trompeurs.

Réciproquement, l'utilisation d'outils de visualisation pour la sécurité nécessite des connaissances en sécurité, en particulier pour analyser des formats de log et des configurations systèmes. Pour faire face à ces situations, les experts en sécurité se fient à leur ressenti pour comprendre la situation, identifier les problèmes, et trouver des solutions. Cette familiarisation se développe en instincts et habitudes qui peuvent finalement devenir de réels protocols.

Dans notre cas, les tâches à accomplir demandent l'exploration de multiples sources d'information différentes, ce qui implique le choix d'un outil spécifique à chaque fois qu'une nouvelle source d'information a besoin d'être explorée.

Compte tenu de ces observations, il semble important de préconiser que la visualisation pour la sécurité doit permettre aux experts de se concentrer le plus possible sur leurs objectifs (la sécurité) tout en les libérant des problèmes en dehors de leur milieu d'expertise (le design).

C'est dans ce but que nous avons conçu ELVis, un outil de visualisation de log pour la sécurité. ELVis permet aux experts en sécurité d'importer des fichiers de log ayant des formats multiples (par exemple, des fichiers de log apache standard et des fichiers syslog tels que les fichiers d'authentification) et de les explorer grace à des représentations pertinentes sélectionnées et générées automatiquement en fonction des données qui ont étés choisies.

Pour offrir ces fonctionnalités, ELVis identifie le format du log, transforme chaque ligne en champs et associe à chacun d'entre eux un type de données. Ces types permettent d'une part d'enrichir les données avec des calculs descriptifs et d'autre part d'extrapoler des champs complémentaires. Ils permettent aussi faire d'associer automatique des visualisations appropriées aux champs sélectionnés.

⋈

L'analyse peut être considérée comme un processus de filtrage progressif que l'analyste adopte pour rechercher d'un élément spécifique d'information. Cependant, même si la détection de chaque action malveillante est fondamentale, il est aussi important de comprendre les relations entre les événements de sécurité pour pouvoir reconstruire le scénario global.

Une fois que l'analyste a trouvé un événement intéressant, il ou elle doit pouvoir découvrir les éléments connexes, même si ces événements se trouvent dans des fichiers de log différents, générés par différentes sources et donc exhibant des formats différents.

Quels sont, par exemple, les relations entre les attaques dans différents composants du système? Une fois qu'un serveur web est compromis, est-ce que les attaquants ont ensuite effectué d'autres actions malveillantes dans d'autres endroits du système? Dans ce cas là, quelles sont les conséquences? En réaction à ceci, nous affirmons que l'investigation en informatique est un processus itératif qui permet à l'analyste de facilement pouvoir utiliser les information stockées dans des fichiers de log, même si ceux-ci ne sont pas explicitement liés *a priori*.

C'est pour faciliter cette tâche que nous avons conçu CORGI, un outil web qui aider à explorer plusieurs fichiers log simultanément et qui permet à l'utilisateur de traverser plusieurs ensembles de données en suivant des points d'intérêt dans ces données.

Pour faire ceci de manière fiable, le système de typage utilisé pour associer des visualisations avec des sous-ensembles de données est étendu avec des types sémantiques qui permettent de découvrir des champs connexes entre les fichiers log. CORGI réutilise les capacités de visualisation de ELVis et les étend pour améliorer les possibilités d'exploration.

Le filtrage réactif des données est implémenté pour toutes les vues, et ces vues réagissent de façon synchronisée pour tout ensembles des données suivant l'interaction avec les points d'intérêt stockés. L'interaction utilisateur est une exploration guidée par des points d'intérêt et une interface conçue pour les cycles de recherche avec une approche de filtrages progressifs.

Les points d'intérêt peuvent non-seulement être utilisés pour filtrer et lier plusieurs logs, ils peuvent aussi donner des perspectives sur la progression et les résultats d'une analyse, fournissant ainsi l'essentiel pour partager des sessions et pour automatiquement générer des rapports.

ABSTRACT

Managing the vast quantities of data generated in the context of information system security becomes more difficult every day. Visualisation tools are a solution to help face this challenge. They represent large quantities of data in a synthetic and often aesthetic way to help understand and manipulate them.

In this document, we first present a classification of security visualisation tools according to each of their objectives. These can be one of three: monitoring (following events in real time to identify attacks as early as possible), analysis (the exploration and manipulation *a posteriori* of a an important quantity of data to discover important events) or reporting (representation *a posteriori* of known information in a clear and synthetic fashion to help communication and transmission).

We then present ELVis, a tool capable of representing security events from various sources coherently. ELVis automatically proposes appropriate representations in function of the type of information (time, IP address, port, data volume, etc.). In addition, ELVis can be extended to accept new sources of data.

Lastly, we present CORGI, an successor to ELVIS which allows the simultaneous manipulation of multiple sources of data to correlate them. With the help of CORGI, it is possible to filter security events from a datasource by multiple criteria, which facilitates following events on the currently analysed information systems.

ACKNOWLEDGEMENTS

To Nicolas whose help is too often overlooked. I could not and would not have arrived this far without yours.

To Frederic for being patient with me beyond boundaries. Sorry for the angry outbursts.

To Christophe, who plays the bad guy so well we sometimes forget how much he actually cares.

To Clémence for putting up with me for so long, for helping me to keep a life and for making me smile through tough times.

To Mom for teaching me about kindness, and to Dad for teaching me to think about people.

To Thomas for that extra dose of crazy.

To my friends for asking questions and helping me to answer mine.

Thank you all dearly.

Rennes, October 2015 C.

CONTENTS

INTRODUCTION

> “A common mistake that people make when trying to design something completely foolproof is to underestimate the ingenuity of complete fools."
>
> — Douglas Adams

The increasingly ubiquitous nature of computing has been accompanied by an ever growing need for better and more reliable security.

Although some security issues can be prevented using security mechanisms, complete systems knowledge is often beyond the scope of understanding and grows continuously. To help with unpredicted issues, additional tools are used to detect anomalous activities and known attacks. However, these also require configuration, which again means fully understanding the concerned systems and the vast amounts of data they produce.

In this section, we discuss the proactive security strategies adopted to ensure information system security, why reactive security measures are still required, and what problems these in turn create. We then introduce visualisation as a solution to some of these problems, along with the challenges presented by security visualisation.

INFORMATION SYSTEM SECURITY

Information system security aims to protect information and services against external and internal attackers. It consists in insuring the three following fundamental properties:

- *Confidentiality* consists in ensuring that only authorised parties can access data and services.
- *Integrity* consists in ensuring that only authorised parties can modify data and services.
- *Availability* consists in ensuring that data and services are available to authorised parties when required, even when systems are under attack.

These properties are applied according to a *security policy*, a set of clearly expressed rules which define what is authorised and/or what is forbidden. Confidentiality, integrity and availability can either be ensured proactively or reactively.

Proactive security

Proactive security consists in enforcing the security policy *a priori*.

First, *secure programming* techniques and frameworks allow programmers to develop operating systems and applications with fewer implementation errors. Second, *configuration hardening* consists in tuning the configuration of the various components (OS, applications and protocols) to make a given setup more robust in a given production environment. Third, *authentication* and *access control* restrict access to the system resources to verified and authorised entities, preferably by ensuring the *least privilege* principle. Fourth, communication security, often relying on *cryptography*, ensures their confidentiality and integrity. See [4] for many examples of applications and technologies.

In theory, the tools and procedures used to enforce the security policy for known data and services are designed to guarantee their security. However, in practice, policies, procedures, configurations, tools and protocols can display flaws due in part to incomplete knowledge and/or flawed implementations.

Ensuring security fundamentals is very difficult, as is making security systems unbreakable. The task of taking all scenarios of use and possible flaws into account dramatically increases in difficulty and requirements. The number of users, and their frequency and concurrency of use make future proofing systems a now almost impossible task. This is why, aside from the proactive security solutions that try to ensure *a priori* confidentiality, integrity and availability, reactive security is also used.

Reactive security

Reactive security consists in designing systems to detect, limit and possibly correct compromises.

Reacting to security issues in systems requires maintaining accurate representations and activity histories for every component which composes these systems. The most widely used approach for obtaining this information is logging, which requires a new category of tools capable of distributed collection and analysis of the resulting datasets. The resulting datasets are rarely small, in part due to the size of the systems but also because the archived history needs to be as far reaching in time as possible. Because of these factors, analysis without the assistance of automatic tools is often unfeasible.

Autonomous measures such as antiviral software and monitoring software help to detect misuse of services and anomalies in agent behaviour. Monitoring software helps operators stay in control of systems by detecting changes in metrics and system health, following symptoms which can indicate anomalies and attacks.

Intrusion Detection Systems (IDS) are another category of reactive security tools. They collect information about events on the systems they monitor and try to detect attacks from this information. When an attack is detected, an alert is raised.

IDSes can first be classified according to the type of system they monitor [15]:

- Network Intrusion Detection Systems (NIDS) such as [65] for instance monitor networks.
- Host Intrusion Detection Systems (HIDS) such as [9] for instance monitor operating systems or the interactions between the OS and the applications (often, system calls).
- Application Intrusion Detection Systems (AIDS) such as [64] monitor the internals of applications.

Whatever the source of information they use, IDSes can also be classified according to the approach they follow to detect attacks. Two different approaches currently exist: *misuse-based detection* and *anomaly-based detection*.

Misuse-based intrusion detection systems are loaded with patterns to know what an attack looks like and try to match any event on the monitored system with the attacks they know. They therefore require misuse databases that are as complete and up-to-date as possible. This kind of IDS generally reacts well and reliably to known attacks. However, it misses unknown attacks (especially zero days[1]) since, by definition, these attacks are not in its database, and produces bad results when signatures are too broad or narrow. Therefore, *misuse-based IDSes* potentially suffer from numerous false negatives, i. e. they do not raise alerts when some attacks occur.

In contrast, *anomaly-based* intrusion detection systems know how a network, system or application behaves in normal conditions and react to unusual behaviours that are considered as attacks. Anomaly-based IDSes therefore require a training phase to learn what a *normal behaviour* looks like. Anomaly-based IDSes handle unknown attacks better than misuse-based IDSes as long as these attacks exhibit a behaviour that is significantly different from the one that is considered normal. However, the learning phase is often a problem:

- either it provides a very specific description of the normal behaviour, in which case some slightly different but normal behaviour will trigger an alarm (false positive),
- or the description is very broad and some attacks will go undetected by mimicking a normal behaviour (false negative),

[1] *Zero days* are attack methods that are unknown from the community at a given time, and especially when they are used by attackers for the first time.

- and in both cases, there is no easy way of making sure that the training data is free of attacks.

In practice, both these approaches are used in varying combinations, but training and/or up-to-date databases are still required and both generate false negatives and/or false positives. Specifically, security operators are often overwhelmed by alerts, many of these being redundant. Research is currently underway on tools which help correlate and aggregate alerts to make them more manageable [24]. Nevertheless, it is often difficult for operators to decide which ones are valid. In consequence, extensive human intervention is still required, which makes fully-automated intrusion detection unfeasible or at least imperfect in practice.

BRINGING PEOPLE BACK INTO THE LOOP

Automatic security tools used for reactive security such as intrusion detection systems aim to offer an automatic and fast approach to detect intrusions. They are programmed to respond to situations with known patterns or situations which diverge from normal behaviour, and excel at repeating this action. However, such systems do not react well to unknown events, trigger many false positives and miss important events. Therefore, human operators are still required to make sense of reported security events. Unfortunately, for their interventions to be useful (especially when the task consists in analysing alerts), they need to quickly understand situations, easily obtain overview and answers to questions, and consult a large context of data for reference. For them to perform this in a relevant way, it is necessary to keep a precise record of every event which happened on the monitored system. This record should be as complete as possible, and in most cases, confidentiality, integrity and availability should be insured for them too. This leads to a scalability problem: an ever growing amount of data needs to be stored, indexed, protected and processed, at least partially, by people.

Machines often store human readable text logs of their activities for easier access by operators. These logs tend to be much too large for human friendly reading though. When these logs are processed manually, standard UNIX text processing tools are used, such as `sed`, `grep`, and `find`. While these command-line tools offer a flexible approach to data mining, using them requires a lot of concentration for what is often slow manual data mining. Because of the scales of data, manual data mining is too time and mind-consuming to remain an option. In fact, even automatic solutions sometimes show signs of slowness and fail to perform fast enough [18].

In light of the situation, research in security analytics has recently been increasingly centred around security visualisation. Security visualisation lies between manual and automatic solutions, and aims to combine the best of both into usable tools. In other words, using visual representations experts aim to to improve the monitoring and data mining process for people by helping to better communicate security data.

THE CHALLENGES WITH SECURITY VISUALISATION

Visualisation has historically been used to represent data so ideas can be more easily shared or understood. First used in the form of maps and later with diagrams, the modern science of visualisation is an expansive field. During the last two centuries many techniques and designs have been invented to address new problems [72].

This has lead to sophisticated visual representations of data finding places in day to day life. Charts have become popular, sometimes even taken for granted and often used to display numbers and trends in presentations or infographics. Security and visualisation are not new partners; a modicum of visual representations is now expected when security is concerned.

Although the problems in security are sometimes similar to those in other domains (e.g., monitoring information flow), designing visualisation tools for security is made challenging by the disparity of security issues: data formats and protocols vary, the standards are numerous, hardware and software configurations are often custom [6]. While some domains can rely on the unchanging nature of problems or the availability of time, security issues are often new and require immediate attention by nature.

However, as with any tool, good security visualisation tools need to help experts reduce tedious and repetitive tasks to a minimum, and let them express their expert knowledge as directly and easily as possible.

CONTRIBUTIONS

The published contributions of this thesis are threefold:

In order to better understand the combination of visualisation and security, the approaches taken by research and available tools need to be studied. We develop a point of view based on scenarios involving personas faced with problems and goals, and split security visualisation in general between three goals: monitoring, analysis and reporting.

The second contribution, called ELVis [33], is the proof of concept for a tool which addresses the experience requirements faced by experts needing to visualise security data. Using an application design which separates security knowledge from visualisation knowledge, experts from each party can add their knowledge to the tool, and security practitioners can explore security information without prior training in visualisation.

The third contribution, called CORGI [32], extends these ideas for better exploration of security data. By first helping experts to collect points of interest in logs, they can then traverse multiple sources of data during a single session and also use these points to share information and build reports.

The next chapters broadly follow the contributions in their chronological order of implementation. The first chapter discusses the state of visualisation, transformations of data, combinations of visual variables, the perception of visualisation and the design factors for effective visualisations. Chapter two discusses the application of visualisation in security, the problems they try to solve and the personas we can deduce from these problems. We describe how any security tool addresses these problems and personas, each to certain degrees. Chapter three describes ELVis, the first tool developed during this thesis, and chapter four is dedicated to CORGI, which builds uppon the concepts of ELVis.

1 THE STATE OF VISUALISATION

Visualisation presents information for people and aims to make complex data easier to understand and clear enough to help gain insight.

This chapter offers a decomposition of visualisation into interdependent processes operating at different levels of detail. The next section discusses the pipelines which process raw data into visualisations with intermediate transformation steps. Section1.2 focuses on the composition of these steps and frameworks for composing graphical elements. Section1.3 explores the mechanisms in perception which explain human affinity for visual understanding and why visualisation works. Section1.4 describes goal oriented design for visualisation centred around people and their problems and also approaches for improving visualisation design.

1.1 DATA TRANSFORMATION

Although visualisation has been a long standing tool for communicating information, adequate domain languages for describing the design of a visual representation, its components and the process of visualisation have only been developed in recent decades.

Accurate language is important in this case. Software and visualisations are built using semantically meaningful languages specific to the domain. Thanks to improvements in these areas, the construction of visualisations has gradually become increasingly well defined and useful.

Aiming to build a reliable foundation for the production of quantitative graphics, Wilkinson [78] proposes a formal grammar for specifying visual representations as translations of data into graphical counterparts. Part of the work discusses the processes in visualisation software which transform raw data into a final visual representation, and describes a linear multi-step pipeline for data (Fig 1.1).

First, variables are isolated in the raw data. These are then transformed and combined using algebraic methods. The resulting sets are put to scale and a last mathematical step applies statistical methods to obtain information such as averages and frequencies. The next steps determine representations for each datum, assigning geometry to variables, mapping them to a coordinate system, then adding aesthetic parameters.

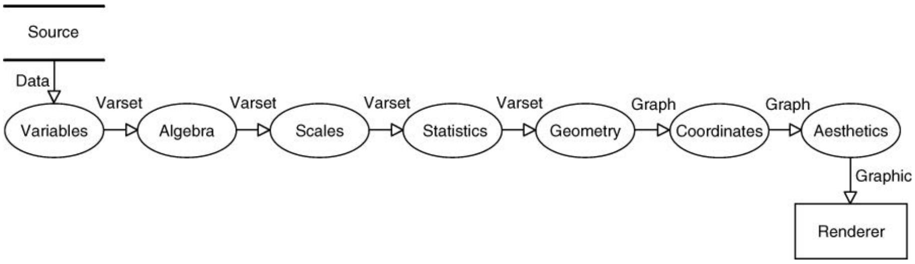


Figure 1.1: Wilkinsons data-flow diagram as pictured in The Grammar of Graphics [78]. Each step provides a different type of transformation, changing the original data incrementally in nature and/or type from the raw data source to final image renderer.

Centred around data transformation, this pipeline models the process of creating a visual representation from raw data. However, the end results are limited to static representations, capable only of communicating information to the user. Building true interactive graphics provides feedback to the system and a direct interface with information, improving the exploration of visualisations.

In contrast, Ben Fry [22] approaches visualisation from a point of view which takes interaction into account. He presents the different aspects of designing fully interactive visualisations. After describing not only the production of graphics but also user interaction and exploration, he describes a pipeline which generalises the execution steps for interactive visualisations (Fig 1.2).

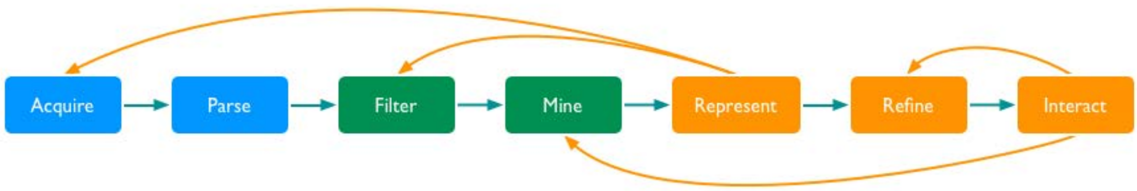


Figure 1.2: Ben Fry's seven stage visualisation pipeline from Visualising Data [22]. Not centred around transforming data, Fry chooses instead to concentrate on the software function at each step. The addition of return points implies a cyclical and interactive mode of operation.

A pipeline describes a process closer to the methodologies of analysis and exploration. The interaction step is meant to represent a basic interface with graphics, but several points of return to previous steps in the pipeline add the possibility for direct feedback from the user. Able to affect changes to the filtering, mining and acquisition steps, the user can change the scope of a view, scale properties and sources of information to influence the end result. This upgrades the visualisation process from a static process to an interactive cycle of visual exploration.

While the information is being interacted with and once the visualisation has been seen, it still has to be understood by the user and transformed from visual representation back into the actual information. To better understand this process, we can study how information is presented and explained in journalism. In [62] multiple narrative structures are described, each with a certain degree of interactivity. In a *martini glass structure* narrative pattern, the user follows an introduction to the subject, which is followed by a tightly guided presentation of the problems, and ends with an open-ended free exploration of the data. The more guided *interactive slideshow* pattern implies a linear presentation of the problem with transitioned steps, where each step allows the user some freedom of exploration to understand it. The *drill-down* pattern is more iterative than the previous two, and is well summarised in the mantra "Overview first, then filter, then details on demand" coined by Shneiderman [63]. The two former techniques rely on one or more steps of bespoke design to construct a narrative. While they are effective for sharing insight and reporting, *drill-down* seems more suited to visual data mining, as it is the closest to the pipeline for interactive graphics. This approach is popular in the security visualisation community, and is closely tied to visual data mining. While the process is not specifically attached to visualisation, it offers an effective approach to visual and interactive exploration of data, while relying on an expressive global visualisation as a starting point for first step understanding.

1.2 VISUAL MAPPING

In the previous section, we introduced grammars which help specify visual representations and the corresponding data pipelines which transform raw data into graphical information. Each step in a pipeline is composed of functions which transform various forms of data and depend on the output of the previous steps. Because specific visual representations can only be built using the right data structures, the functions in each step are chosen according to the requirements of the next step. This implies that each chain of transformations is built for achieving a specific task with specific data.

Even if choosing these functions seems creatively free, they are tightly constrained by the data and the visualisation, which is directly dependant on the cognitive and psychological properties of perception. Correctly exploiting these helps build a visualisation which completes its objectives of communication and interaction.

For example, the implications of expressing quantities are different from those of expressing frequencies. Each provides a different view of information and adequate visualisations will be perceived differently. Time is often expressed as a continuous variable, represented as an additional axis, but time can also be discrete, split between seconds, minutes, hours, days and other units, depending on the viewers objectives. Each approach has an impact on the ease of pattern recognition.

We also need to account for the irregular nature of human perception. Even for basic visualisation, different representations are perceived in different ways, and some are better than others for perceiving differences or distributions. Colour is a widely used visual parameter in communication for differentiation. Contrasting colours can be used to represent state or nature: red for danger, yellow for warnings, and green is the "good" colour. However, with colour only a finite number of categories can be perceived and properly distinguished.

Many other graphical properties can be used, such as position, rotation and alignment to better visualise a property of the data. In [5], Bertin gathers his experience in cartography and charting into a synthesis of principles for graphical communication. Using comparative studies of techniques for making use of shape, orientation, colour, texture, volume and size, he constructed a classification of *visual variables* ordered by versatility (Fig 1.3). Following this classification, the recommended best practice is to encode the two most important dimensions to the position of marks, and then assign any further dimensions to the next variables following its order and particularities.

Building effective visual representations of data implies choosing effective graphical techniques, and choices like these ultimately define the tasks a visualisation tool is adapted for. These should therefore be determined according to the objectives and the overall motivations for designing the visualisation. When we wish to convey an effective comparison of different event frequencies for example, using texture would be an unfavourable choice of technique, as it is unsuited for comparing quantities.

To simplify this choice, Mackinlay [48] describes more meaningful and contextual rules for making visualisation design a more automatic process. He proposes a variation on Bertin's visual variables which encompasses more graphical techniques and adds a classification by type of data (Fig 1.4).

These categorisations of graphical techniques are essential for the foundation of meaningful and effective design of information representations. Using visual variables also favours better consistency in design across different representations and enables their use in programs with automatic design. For example, one can automatically determine the number of categories in a visualisation and determine

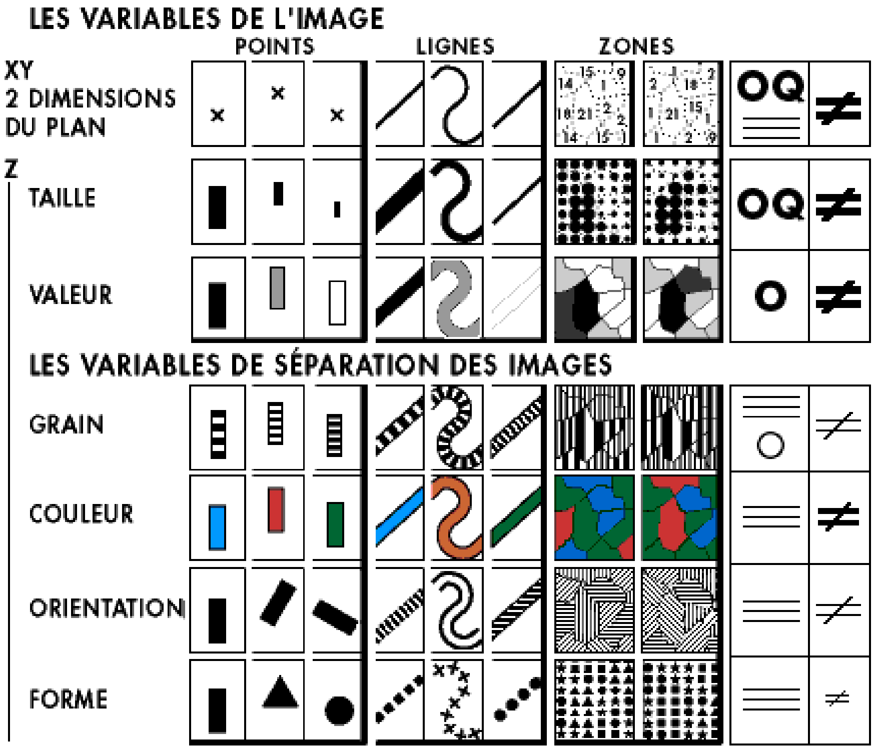


Figure 1.3: The visual variables proposed by Bertin[5]. These are ordered from top to bottom by versatility. Each is more or less effective at expressing the similarities (≡) or differences (≠) between marks, as well as their order, quantifiably (Q) or not (O).

which dimension is best suited for the data. Orientation can be effective for discerning a small number of values, but as a first choice, length, size or position will often be more efficient. This explains the clarity of bar charts compared with the more popular pie chart which can be difficult to understand. When other choices are not available, colour is often used but is much less effective for multiple categories. In any case, we must note that in order for these variables to be effective, they have to relate semantically to the domain the data is from.

Finally, the effectiveness of each variable does not change according to the nature of the data, but because of the particularities of human perception, which are discussed in the next section.

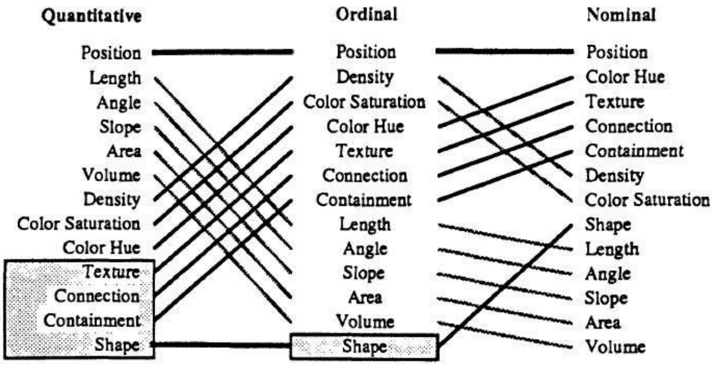


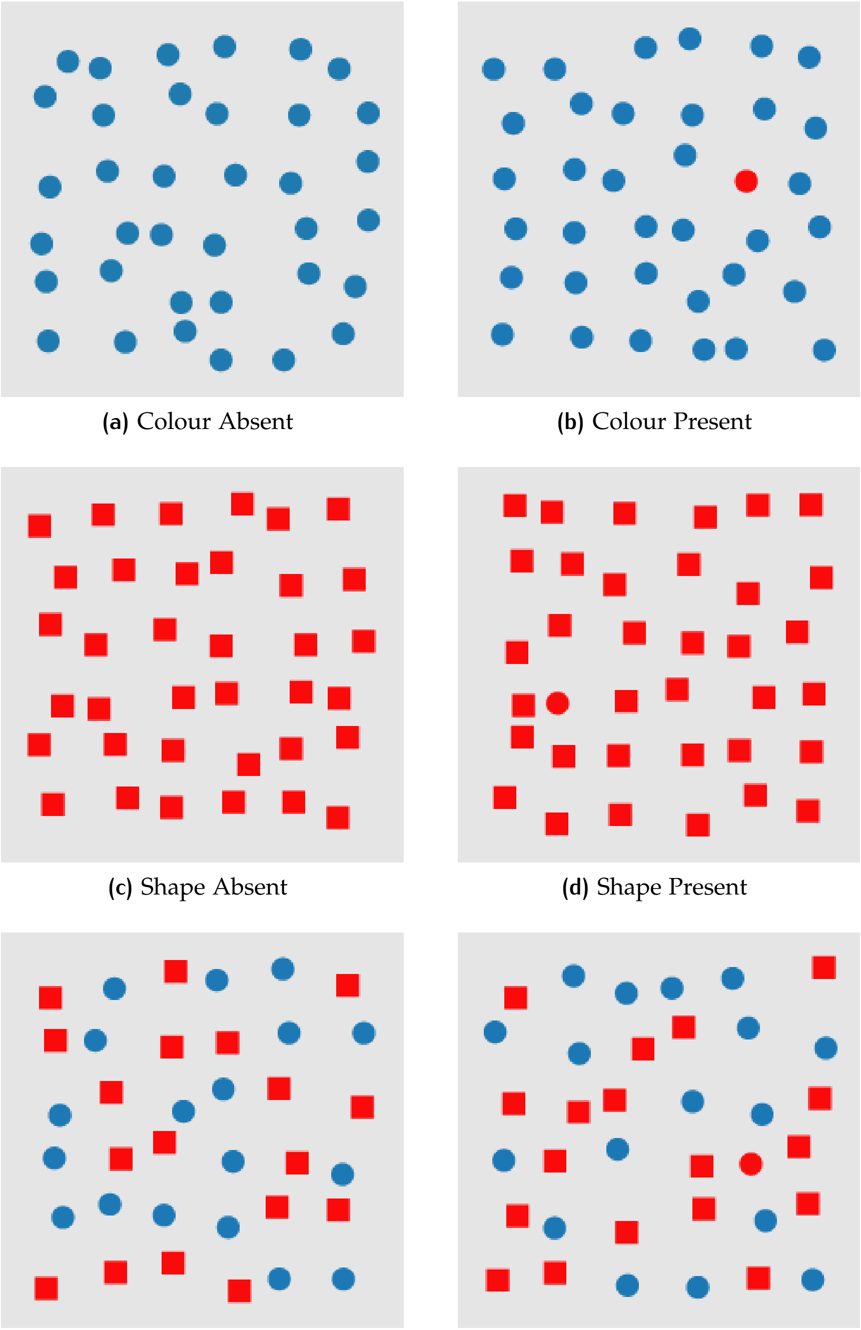
Figure 1.4: Mackinlay extends the set of visual variables [48]. These are listed by effectiveness for three types of data (quantitative, ordinal or nominal). The slope graph illustrates how the effectiveness of a visual variable changes according to the data type. For example, comparing quantitative data by encoding to length is the second most effective approach, but using length to compare ordinal data is much less effective. In this case density is a much better choice of variable. The variables in the gray boxes are not relevant to that type of data.

1.3 PERCEPTION

Human vision has many features which, through natural selection, have evolved according to success factors such as hunting prouesse and danger evasion. A consequence of this process is the variability and inconsistency of perception quality:

- A focal point allows people to focus on elements to inspect them in detail, but peripheral vision is adapted to reacting to movement;
- Some colours such as red and blue, are perceived more clearly than others.
- Counting small groups of marks is almost pre-attentive, but counting large groups requires effort.
- People can recognise familiar shapes and identify objects, but also see these in clouds and stars.

Some visual changes trigger the use of more attention than others and follow a logic similar to Bertin's classification of *visual variables*: the amplitude and nature of the reaction to visual changes are linked to the nature of the visual properties which are changing. A small change in position or size is often easier to notice than a gradual change of colour or texture.



(a) Colour Absent

(b) Colour Present

(c) Shape Absent

(d) Shape Present

(e) Conjoined Absent

(f) Conjoined Present

Figure 1.5: Pre-attentive properties compared and combined.

In addition to their effectiveness at provoking reaction, some visual variables have *pre-attentive properties* [28]. These properties trigger rapid subconscious processing capacities which allow people to understand parts of a situation before being consciously aware of its information (Fig 1.5). By definition, these pre-attentive reactions generally happen within 200 ms and help people detect minute changes in image details.

Strategic use of pre-attentive triggers can signal changes in one element among hundreds, sometimes thousands of others and drastically decrease reaction times [29]. To this end, they are an unavoidable resource when designing monitoring and real-time visualisations which require fast pattern detection.

When pre-attentive reactions are triggered in several locations at once, or when multiple properties are used in combination, the reaction is slower: attention is being called to multiple locations using different evaluation types at once, and the effect is diminished.

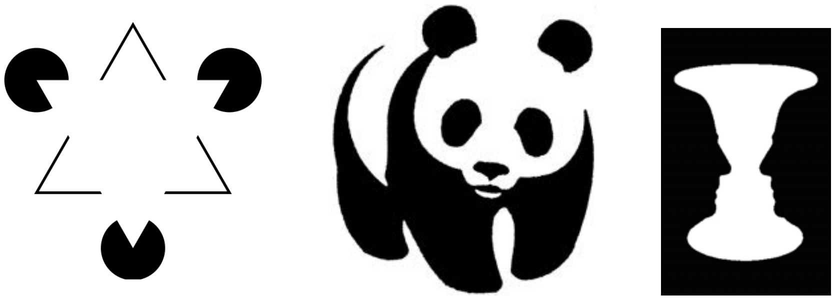
The human capacity to make sense of apparent visual chaos is addressed by the theory of Gestalt[2] psychology. According to this theory, any visual representation is interpreted as a whole, and not merely the sum of its elements. Gestalt theory has four basic properties (with examples in Fig 1.6):

emergence, whereby objects are recognised in their entirety instead of being reconstituted from their components.

reification, whereby shapes are perceived, even when they are hidden or absent.

multistability, whereby images can be simultaneously interpreted in multiple ways.

invariance, whereby objects are recognised regardless of transformation, e.g., size, rotation, position and even deformations.



(a) Kanizsa triangle

(b) World Wide Fund for Nature (WWF) logo

(c) Rubin vase

Figure 1.6: These three figures demonstrate multiple Gestalt properties.
(a) is composed of six distinct shapes, but an extra triangle is perceived in the negative space left by these. This has the effect of completing the discs and back triangle
(b) similarly, although it is incomplete, we instantly recognise the shape of a giant panda.
(c) this image can be simultaneously interpreted both as a vase and as two faces.

[2] German term meaning “shape, form”.

In addition to these principles, the Gestalt laws of grouping explain why humans see certain objects as groups. Objects are subconsciously associated when their visual properties are similar, or when they are close or participate to an optical flow (Fig 1.7). These principles explain why humans still have an edge for extracting information from images when compared to machine learning algorithms. Data visualisation makes use of these phenomena to help people extract patterns, groups and outliers from large amounts of otherwise intractable data in search of possible meaning and interpretation[3].

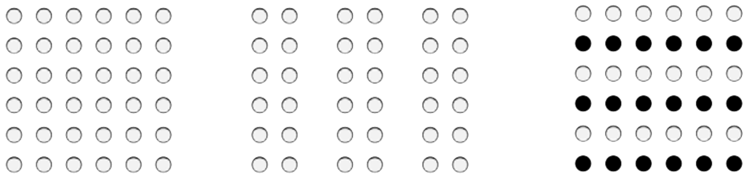


Figure 1.7: Due to the layout of the discs on the left, they are perceived as belonging to the same square. However, when they are split into three columns the viewer sees three three distinct groups. The same effect can be obtained using colour.

However, misuse of these phenomena can also lead to the recognition of coincidental and meaningless patterns. Constellations are named by cultures according to the familiar shapes they seem to form and people often recognise shapes in cloud formations, but no there is scientific basis for predictions made based on these observations. In fact, machines can also succombe to these kinds of false positives: artificial neural nets trained to recognise certain images can demonstrate this phenomenon on a much grander scale [50]. In order to avoiding these false positives, the design must be carefully tested and be accompanied by enough context to debunk misleading patterns.

Changes in isolated or combined pre-attentive properties combined with Gestalt properties help people to quickly detect and even understand correlations or the beginnings of a phenomena in data which machines would have difficulty finding. Together, they partly explain why certain visual variables and their combinations have a different impact than others. By building a visualisation out of marks with attributes depending directly on data, we give a means to the human mind. One such example in visualisation is the heat map or density map. Adjacent points with a single or multiple colours are no longer only linked by proximity, but also form zones which can in some cases be interesting for the user. Similarly, maps representing wind speed using vectors show not only the speed at each point of a map, they also display the shape of wind flow across the entire map (Fig 1.8).

[3] Or as evolution would have it, in search of danger or prey.

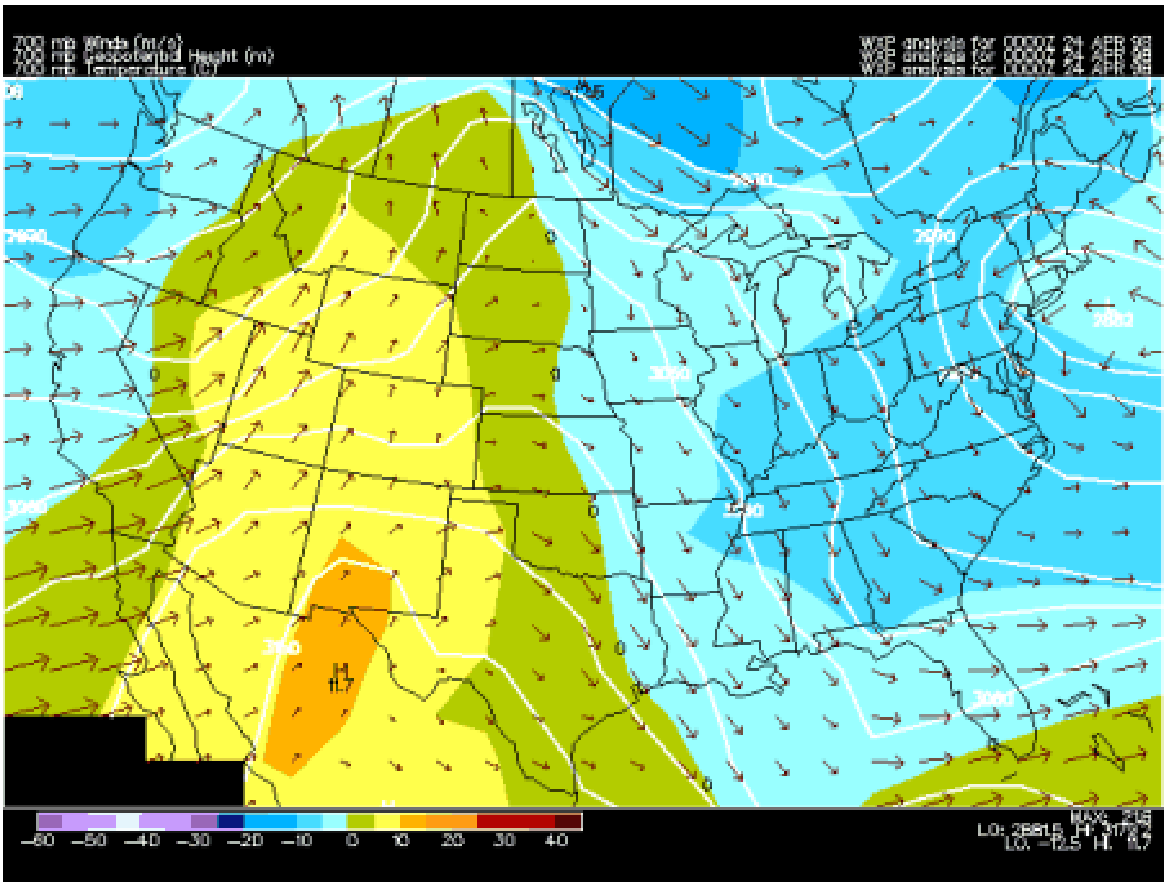


Figure 1.8: Using vector fields, a prime example of optical flow, winds can be represented superimposed on a map already containing temperature zones and state delimitations. **Source:** Department of Atmospheric Sciences (DAS) at the University of Illinois at Urbana-Champaign

1.4 DESIGN

The creation of tools which accomplish their objectives optimally is guided by design principles. Although good marketing may help persuade people to try tools, good design is at the heart of those which are adopted and used on a daily basis. They help people to simplify tasks and accomplish them in an enjoyable way.

Building better tools means taking user constraints into account and carefully understanding the goals of the design to build guidelines. These tools need to be tested and evaluated at every step of the development. This is true for all cognitive objects: languages, games, physical tools, user interfaces, visualisations. . .

Evaluating a design is difficult. There currently exists no single rule which on its own can guide the construction or verification of a tool. All through the development and design processes, multiple factors need to be taken into account to guide designers and provide them with the ability to verify the correctness of the end result.

The design process for a tool needs to take into account the people who will end up using the tool. The challenges of designing around users and the results of failing to do so are already well studied [52].

Goal-oriented design centres the design process around people by aiming to satisfy the needs of the product end-users. To this end, full understanding of the goals of a user needs to be achieved in order to best solve the problem and validate the orientation of the approach.

Goal-oriented design calls for the definition of *personas*, artificial user profiles created after interviewing the real target users. A full persona comprises both the description of the users and their goals by creating names and back stories which correspond to the real users. These personas are artificial composites, but accurately represent real people. Back stories and names help make the personas believable so argumentation is possible while avoiding the personal biases of real people.

1.4.1 Guidelines for effective visual representations

Effective design is made easier when following guidelines. Tufte introduced good practices and rules for the production of effective and truthful data graphics in scientific literature and the media. In a four part series of books [72]–[75] he explores guidelines for successful visualisation by making extensive use of examples.

Among the first guidelines for improving visual representations is the data-ink ratio:

$$\text{Data-ink ratio} = \frac{\text{Data-ink}}{\text{Total ink used to print the graphic}} \tag{1.1}$$

The ratio between the quantity of "ink" directly used to represent data and the total quantity of "ink" used for the representation helps to evaluate the amount of misused space. Maximising this ratio reduces the amount of noise, which can be misleading or confusing, and improves focus and readability of the end product (Fig 1.9).

Another guiding ratio is the *lie factor* between the visual representation and the data itself, this time addressing the perceived amplitude of a variable in its graphical representation and its actual value:

$$\text{Lie Factor} = \frac{\text{size of effect shown in graphic}}{\text{size of effect in data}} \tag{1.2}$$

$$\text{size of effect} = \frac{|\,\text{second value - first value}\,|}{\text{first value}} \tag{1.3}$$

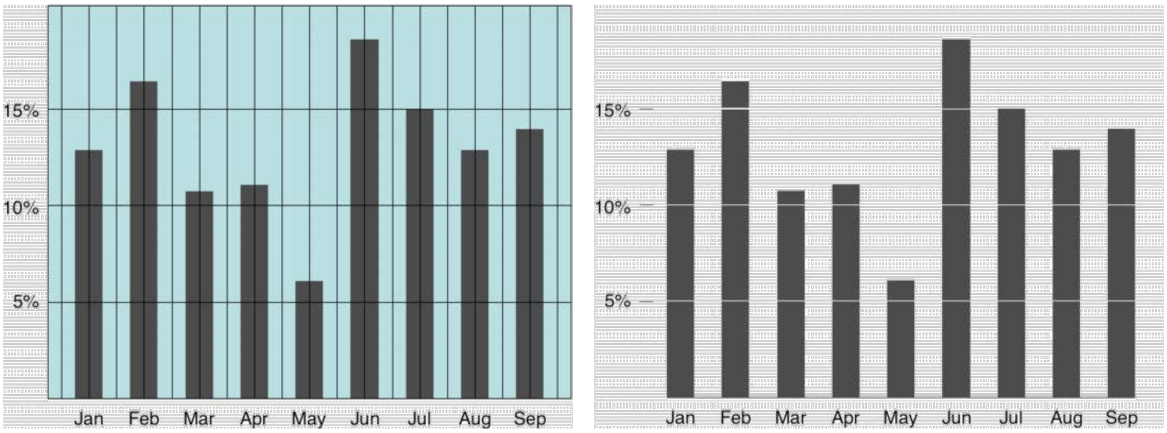


Figure 1.9: In [74], Tufte removes unwanted noise from a bar graph to improve its data-ink ratio.

While this factor is meant more for custom made visual representations, the best practice is to keep this ratio as close to 1:1 as possible. Using non-linear scales and magnification techniques can distort perception of data values for example (Fig 1.10).

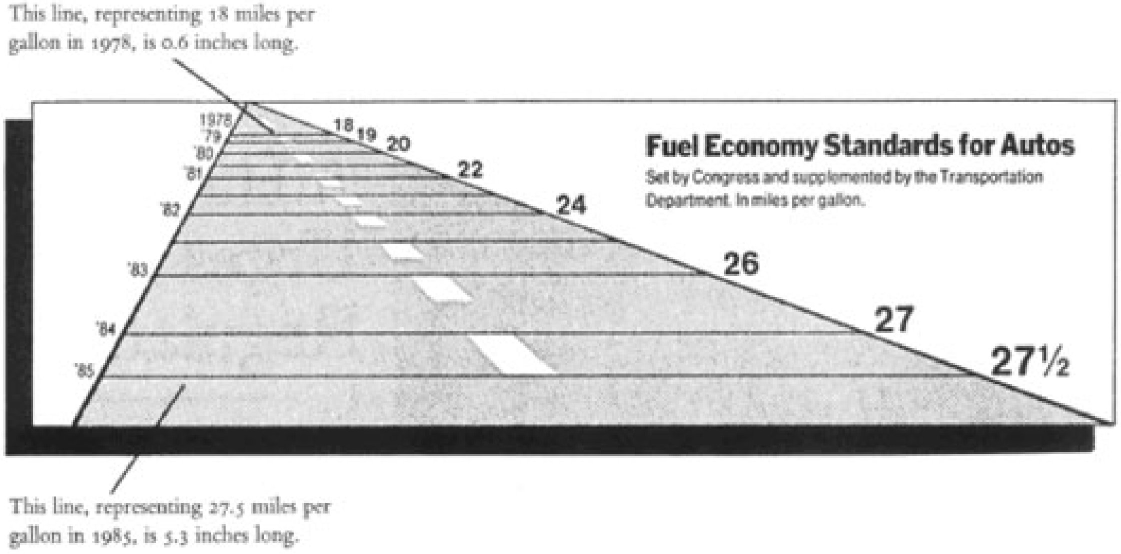


Figure 1.10: In this graphic, originally published by the NY Times, perspective is misused to exaggerate the growth of the evolution of mandated fuel economy standards for autos set by the US Department of Transportation [74]. The data shows an increase of less than 200%, while the image displays an increase of almost 900%.

One way of addressing these two factors is to reduce "chart junk", the needless decorative aesthetics often encountered in visual representations. Again, this issue largely concerns single purpose diagrams and infographics. However, care should be taken when including

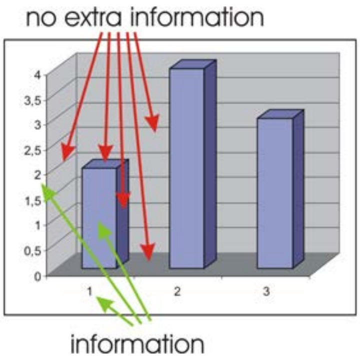


Figure 1.11: Displaying these bars in 3D is confusing and unnecessary chartjunk, as are the guides and surfaces which provide no extra information.
Source: Leonhard Seyfang, 2005

these extra visual artefacts, even when they provide visual references or redundancy (see Fig 1.11).

$$\text{data density of display} = \frac{\text{number of entries in dataset}}{\text{area of data display}} \tag{1.4}$$

1.4.2 Cognitive dimensions of notations

The cognitive dimensions framework[7], [8] helps to guide and evaluate the design process. Cognitive dimensions are designed to be a light means of evaluating design quality, instead of relying of in-depth analysis. They offer a shared vocabulary for discussing factors when designing the interfaces for objects.

Cognitive dimensions provide an abstract, high level description of user interfaces and how users interact with them. It deals with them in terms of consistency, error-proneness, hard mental operations, viscosity or premature commitment. These viewpoints help with the creation of new interfaces based on already existing designs by repositioning the design on a particular dimension.

Cognitive dimensions are factors which can help people evaluate usability and design of tools. These dimensions help with discussing and building new tools based on the flaws in others. They are used not only for user interfaces but for all cognitive interfaces: programming languages, visualisations, diagrams...

Green's original list [26] included 14 different cognitive dimensions, which can be extended depending on the domain:

abstraction gradient: To which extent does the notation abstract the problem?

closeness of mapping: How close is the notation to the problem?

consistency: When part of the notation has been understood, can the rest be successfully guessed?

diffuseness / terseness: How efficient is the notation at producing the desired result or meaning?

error-proneness: Is the notation likely to influence the user into making a mistake?

hard mental operations: How much mental work is required at the notational level, rather than at the semantic level? Does the user need to resort to taking notes for keeping track of things?

hidden dependencies: Are dependencies between entities in the notation visible? Are these dependencies indicated in both directions? Do changes in one area of the notation lead to unexpected consequences?

juxtaposability: Can different parts of the notation be compared side-by-side at the same time?

premature commitment: Do tasks need to be accomplished in a specific order? Are there decisions that must be made before all the necessary information is available? Can those decisions be reversed or corrected later?

progressive evaluation: To which extent is it possible to evaluate or obtain feedback on an incomplete solution?

role-expressiveness: How obvious is the role of each component in the notation as a whole?

secondary notation and escape from formalism: Can notations carry extra information outside of syntax, such as layout, color, or other cues?

viscosity: Are there any barriers to change? How much effort is required to make a change using the notation?

> ***'knock-on viscosity'***: changes in the code violate internal constraints in the program, whose resolutions may violate further internal constraints.
>
> ***'repetition viscosity'***: a single action within the user's conceptual model requires many, repetitive device actions.
>
> ***'scope viscosity'***: changes to the size of the data set require changes to the program structure itself.

visibility: How easily can parts of the notation be identified, accessed and made visible?

Not all of these guidelines apply to visual representations. Some are meant to be applied to more abstract cognitive tools such as languages. They are also not designed to be followed as strict notational rules, and are therefore difficult to implement programmatically. They are nevertheless useful criteria for broadly evaluating a tool, isolating flaws and finding starting points for discussing the issues.

1.5 CONCLUSION

In this chapter, we have addressed visualisation as a medium of transforming data into visual information and briefly discussed transformation pipelines. We have also discussed how visual representations can be broken down into visual variables, directly associated by mathematical variables in the data. We have seen how their individual effectiveness changes when facing different types of data and different problems. We followed with a brief study of pre-attentive processing and gestalt perception to understand how we perceive combinations of marks and help choose variables and layout. The last part first addressed visualisation design around goals and personas, with tools such as cognitive dimensions of notations to help evaluate design decisions, and ended with broad guidelines for improving the clarity and richness of visualisations by removing misleading visuals and increasing data density.

In the next chapter, we build a viewpoint on the state of security visualisation by studying published research literature and security visualisation tools. By examining and categorising the applied techniques, we gain a better understanding of which problems are being solved and how.

2 VISUALISATION FOR SECURITY

> "An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem."
>
> — John Tukey

Research in security visualisation has seen a gradual increase over the last decade, and visualisations now frequently feature in security tools both on the silver screen and in real-world security situations.

Using visualisation to display information in large quantities is effective for explaining the complexity of a problem. The effect is convincing and useful for relating security issues to the public. However, the goal of this thesis is to apply visualisation to tools which help security practitioners understand security data.

Security visualisation is often influenced by visualisation from other domains, such as biology. These techniques can sometimes be directly re-applied to problems in security where the data has similar characteristics. For example, biological visualisation tools such as Circos are used to analyse genomic data but have also been applied to situational awareness [21], [46] and communication analysis [34] (Fig 2.1). Hive plots [31] have been advocated as an alternative to node-link graphs for representing protein transfers in bacteria. Because this data largely resembles the function calls in operating systems and software, the same visualisation techniques have been studied for visualising distributed memory computations [17].

As this work focuses on security events and network security, the scope of this thesis will be limited to cases where time is a factor, putting aside other fields such as static program analysis and malware structure visualisation for example. Currently visualisation for security events follows an approach and objectives adopted by teams tasked with the reactive security of systems, and serves at least one of three purposes:

monitoring Using dashboards of visualisations to monitor systems and networks, operators make sure that metrics are within optimal ranges to guaranty service availability and watch for the patterns for known attacks and malicious behaviour.

analysis When anomalies are detected by operators using monitoring software or when systems fail altogether, visual analysis

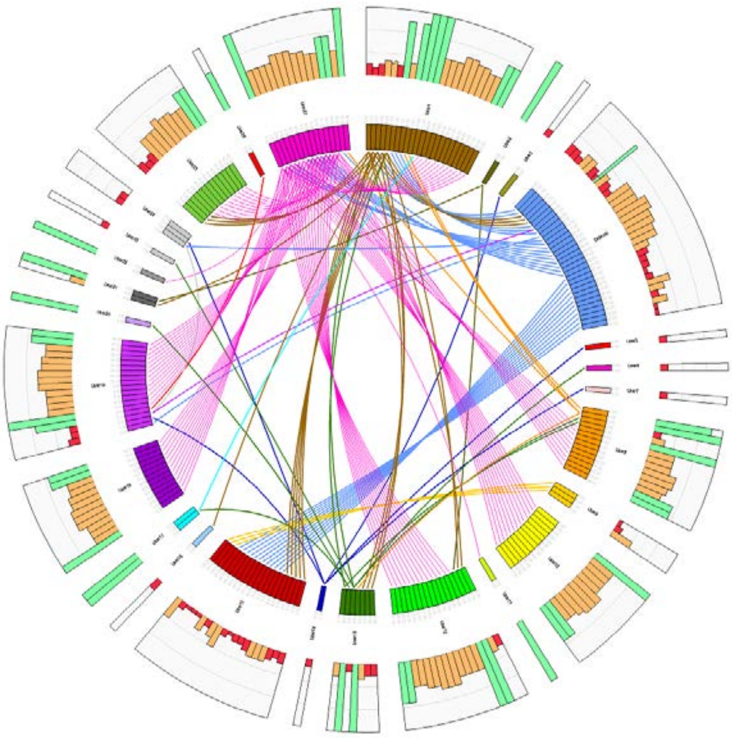


Figure 2.1: Using Circos to visualise inappropriate email circulation on a corporate network [34]. Each ring section represents a single colour coded user. Central connections indicate sent emails, colour coded by sender, and external histograms indicate period (red is recent, orange less, and green is old). The blue user is rapidly identified as a source and the magenta user as a mass emailer.

tools are used to explore and analyse security data. Their objective is to find explanations for incidents, recreate attack scenarios and characterise intrusions missed during monitoring.

reporting When an attack scenario is ready or malicious patterns are discovered, these insights and hypotheses need to be passed on to others for the next courses of action. These reports are generated both as part of routine protocol but also in times of incident when people with higher executive authority need to intervene.

These three purposes are embodied by three personas [66] which represent individuals facing security information (Fig 2.2).

These personas represent the states of people within an operational security team. Some are monitoring the current system states, while others analyse security data to understand new attacks and find undetected others. Every team member needs to collaborate with others to cover the tasks as best as possible, and can communicate effectively through reporting, both inside the team and with external parties.

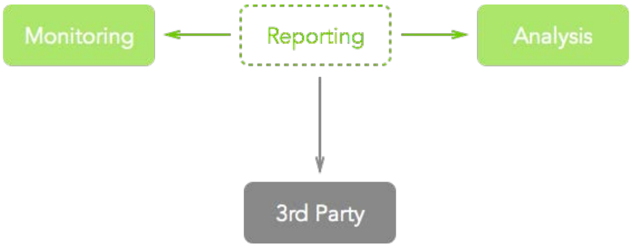


Figure 2.2: A team of three personas. Monitoring and analysis personas communicate with each other and third parties by temporarily assuming a reporting persona.

A *monitoring persona* is tasked with watching realtime feeds of information. The collected information represents the current situation of systems and networks as closely as possible to reality and is therefore time indexed and displayed in real time. While this information can be obtained raw and unprocessed, it will more often be preprocessed by other systems tasked themselves with detecting anomalies and intrusions (as stated in the introduction). As these systems are not perfect their task can be regarded as a reduction of the work load and an initial filter step. The operator is tasked with watching for anomalies in case of false negatives and validating alerts in case of false positives. Due to the real time nature of the task, most of the patterns for normal or anomalous activity need to already be known, and the visualisations are preconfigured to help with catching these patterns. An operator in this situation has little or no time for referring to external sources of information and can only rely on personal experience to understand situations and react as fast possible with effective decisions. Not only is the information delivered in real time, but systems and networks also continue to increase in size. Operators need to monitor continuously increasing amounts of information and coordinate with other operators in order to manage sometimes unforeseen and punctual spikes of possibly legitimate traffic.

An *analysis persona* is tasked with analysing historical information to reconstruct scenarios for unexplained anomalies found during monitoring phases and to find anomalies undetected by previous systems. Data mining is time consuming, but reacting to situations is less crucial when analysing security data than when monitoring systems. Operators require flexible visualisations for manipulating their view of the information, filter out unwanted data to focus on specific sets information and concentrate on details. As the information is historical, operators are also able to explore the data across time. This is crucial to reconstructing events and understanding new scenarios. Other sources of information may be imported and referenced to provide context and enrich information. The exploration process itself is an invaluable source of reference to the operator, who frequently takes notes to document progress and not lose track.

A *reporting persona* can follow one of the two previous personas who has found anomalies or completed the analysis of new information. Following the discovery of this new information, collaborating with other people is often necessary. External parties have the expertise required to understand the situation or executive authority required to react. In both cases anyone receiving these reports needs to understand the reconstructed events. This can be made easier through annotation and recommendation from the operator. The context of the events and how they were discovered need to be comprehensively and equally understood by all parties so that all misunderstandings are avoided.

In this chapter we will study security visualisation tools from the perspective of these three points of view. The next three sections will start with descriptions of the problems and requirements of a persona tasked with one of the purposes. We will then describe techniques used and shared by security visualisation tools designed to solve each of these problems.

2.1 VISUALISATION FOR MONITORING

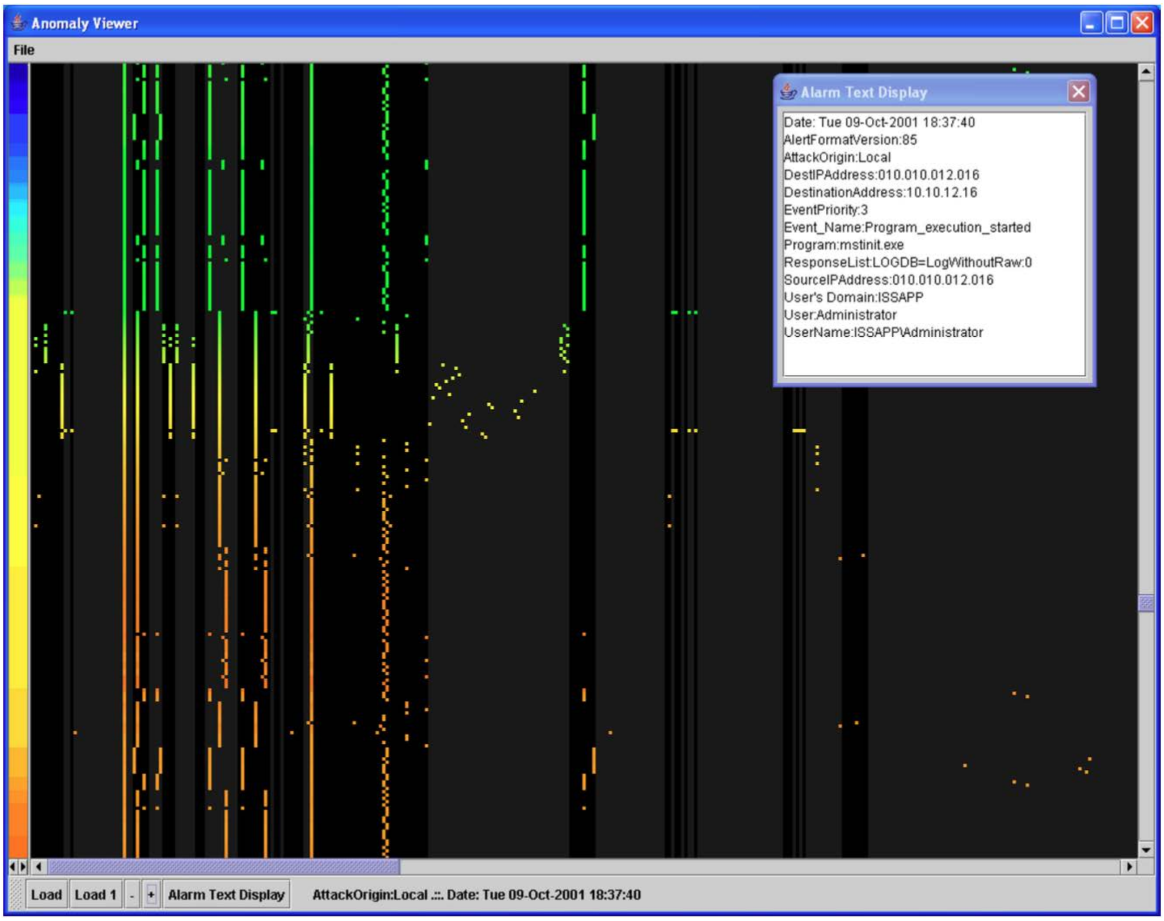
When monitoring systems and networks, operators use security visualisation to more easily cover as many systems and important factors as possible. Visual representations of system states and network communications allow the operators to detect significant change in real-time and detect anomalous behaviour patterns without being inundated by alarms or relying on potentially mis-trained detection systems. These visualisations are designed to solve specific problems with data, and the right configuration of visual variables can help operators catch the right patterns.

Dashboards generally use arrangements of visualisations designed to effectively display metrics along with their bounds and history. To catch more evasive traffic patterns and make use of correlation, more complex configurations and visual representations are designed, but generally monitoring tools are designed to solve a subset of problems depending on the situation. By design, they are not often flexible outside of these considerations.

In order to function well in production, monitoring visualisations are optimised for three distinct features: familiar pattern recognition, situational awareness and scalability.

2.1.1 Familiar patterns

As discussed in the introduction, operators rely on familiar visual patterns and watch for outliers in these which could indicate a change in events.



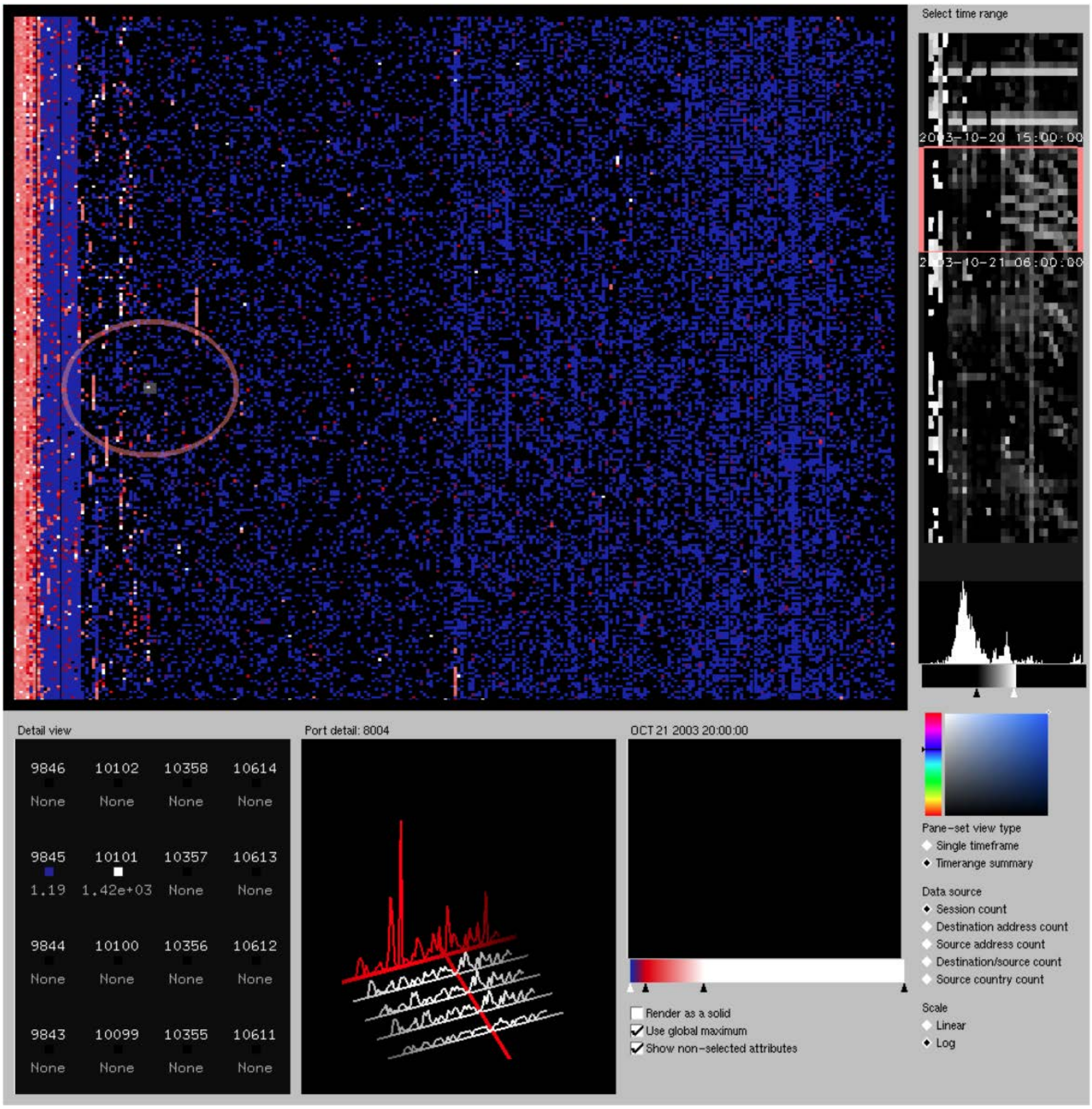
```
Date: Tue 09-Oct-2001 18:37:40
AlertFormatVersion:85
AttackOrigin:Local
DestIPAddress:010.010.012.016
DestinationAddress:10.10.12.16
EventPriority:3
Event_Name:Program_execution_started
Program:mstinit.exe
ResponseList:LOGDB=LogWithoutRaw:0
SourceIPAddress:010.010.012.016
User's Domain:ISSAPP
User:Administrator
UserName:ISSAPP\Administrator
```

AttackOrigin:Local .::. Date: Tue 09-Oct-2001 18:37:40

Figure 2.3: To search for patterns in RealSecure[59] alarms, Colombe *et al.* display[12] each alarm in time as a row of properties and colour coded by relevance.

In order to detect patterns in events, tools often provide scatterplots[4] for simple two dimensional correlation. By selecting the right dimensions to correlate, operators who understand the normal function of systems will recognise when patterns for those dimensions diverge. Colombe *et al.* [12] watch for anomalies in RealSecure [59] alarms by correlating the properties of each alarm in detail (Fig 2.3). PortVis [49] maps events by port and time as single pixels in an effort to maximise data density and concentrate on detecting port scan activity by displaying patterns in time. This task is split between two scatterplots, using one to cover all ports for a given period and another as a port activity timeline (Fig 2.4). IDSRainstorm [1] aggregates addresses and time to display multiple class B address spaces for one day of traffic (Fig 2.5). IPMatrix [38] visualises the sources of Snort alarms by splitting the IP addresses by octet and visualising

[4] a graph in which the values of two variables are plotted along two axes, the pattern of the resulting points revealing any correlation present



9846	10102	10358	10614
None	None	None	None
9845	10101	10357	10613
1.19	1.42e+03	None	None
9844	10100	10356	10612
None	None	None	None
9843	10099	10355	10611
None	None	None	None

□ Render as a solid
☑ Use global maximum
☑ Show non-selected attributes

Pane-set view type
○ Single timeframe
● Timerange summary

Data source
● Session count
○ Destination address count
○ Source address count
○ Destination/source count
○ Source country count

Scale
○ Linear
● Log

Figure 2.4: PortVis [49] tool makes use of two large scatterplots. The left one displays port usage for the entire 65536 port range by mapping traffic within a 256 by 256 coordinate system. The other displays traffic according to time for the entire dataset timeline and enables filtering by time period and the detection of time related patterns such as port scans.

local and internet level activity in two scatterplot visualisations (Fig 2.6).

Scatterplots are effective tools for visualising correlations between two dimensions, as the two main dimensions are encoded to the position of each mark, the most versatile visual variable (See Fig 1.3, page 11). With marks as small as pixels maximum data density can be achieved, and further dimensions can be added to the visualisation using visual variables such as colour or shape for categorical data and size or value for ordinal data. However these extra dimensions are

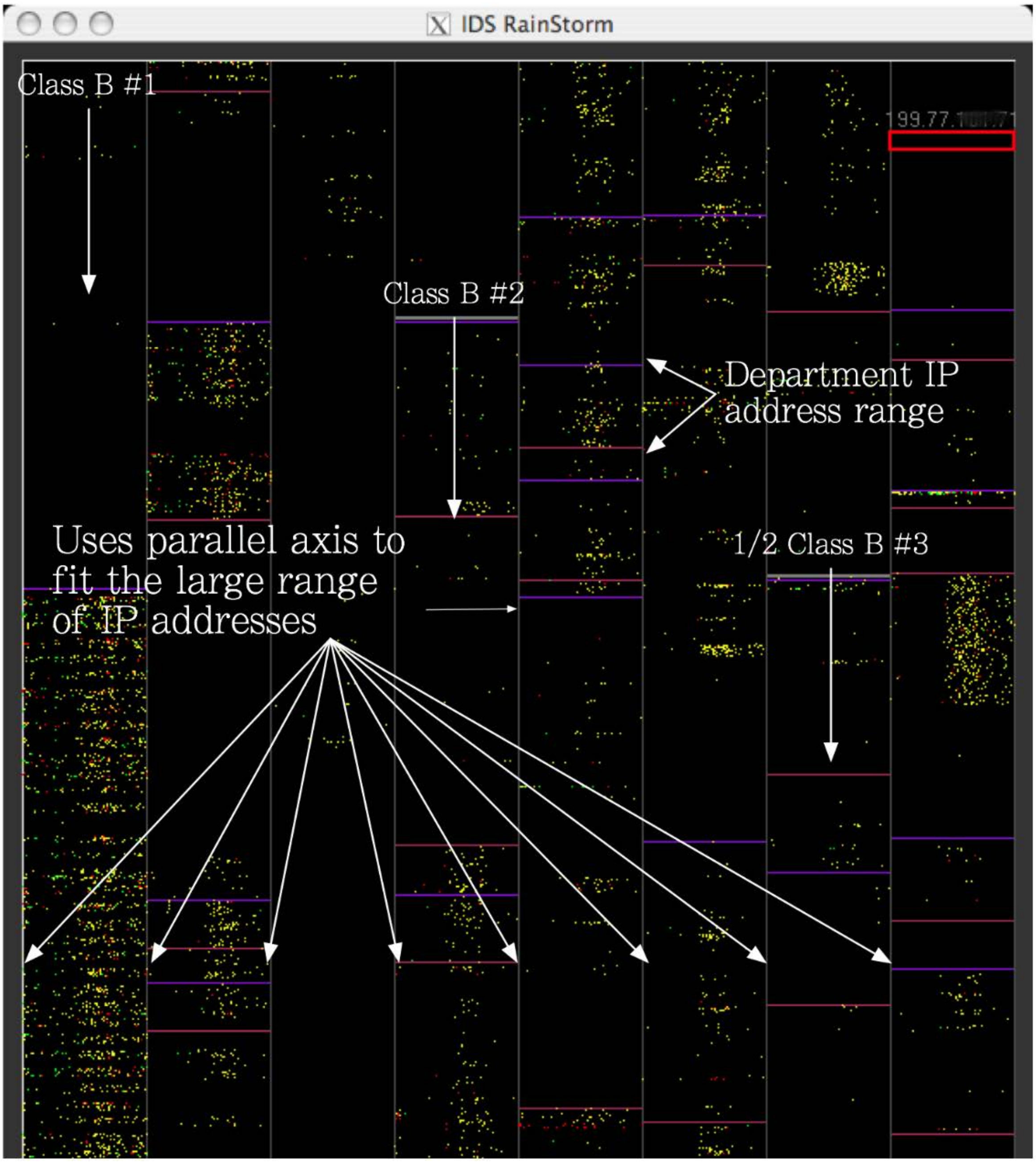
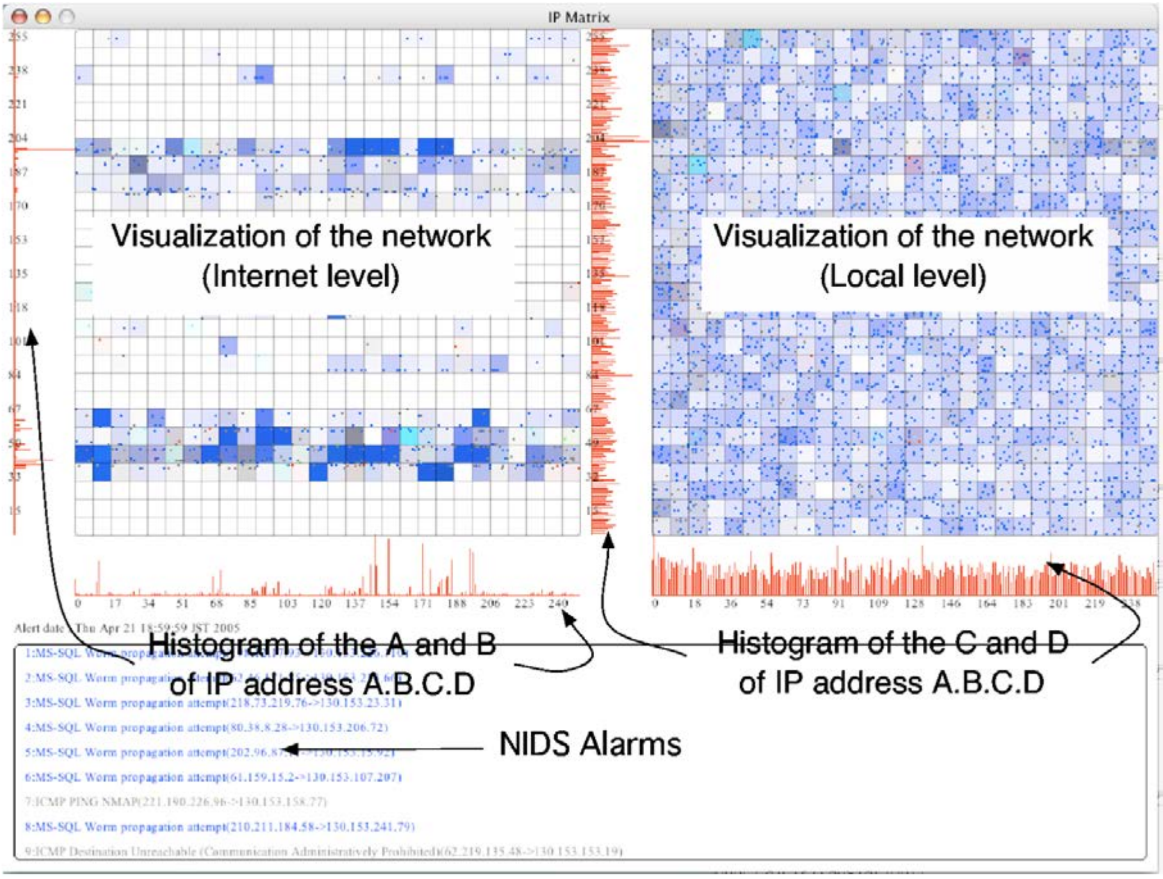

IDS RainStorm
Class B #1
Class B #2
Department IP address range
199.77.
Uses parallel axis to fit the large range of IP addresses
1/2 Class B #3

Figure 2.5: IDSRainstorm [1] displays an entire days worth of alerts on the GeorgiaTech network. This first main visualisation is arranged into columns to represent several class B address spaces, each of which displays a day of alerts along the horizontal axis as rows of pixels, colour coded by severity. The vertical axis is used to group addresses, each representing twenty addresses. A pixel aggregates several alerts and addresses and represents the most severe alert in that group of addresses for that time.

severely limited in distinguishable values compared with the range available for the position variable. This constraint makes visualising patterns for more than two dimensions difficult, particularly when time is considered.



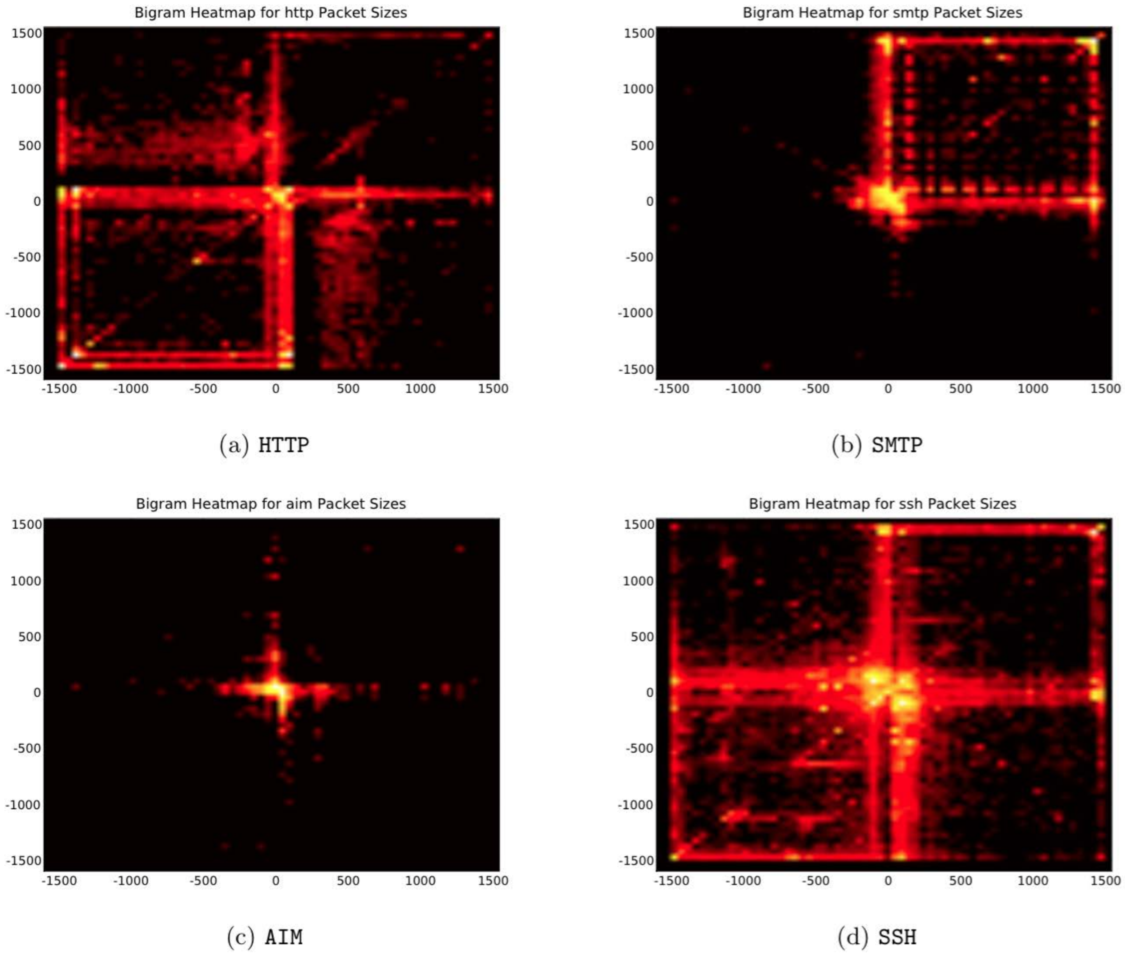
Alert date : Thu Apr 21 18:59:59 JST 2005

```
1:MS-SQL Worm propagation attempt([illegible])
2:MS-SQL Worm propagation attempt(62.[illegible]->130.153.2[illegible].60)
3:MS-SQL Worm propagation attempt(218.73.219.76->130.153.23.31)
4:MS-SQL Worm propagation attempt(80.38.8.28->130.153.206.72)
5:MS-SQL Worm propagation attempt(202.96.87.[illegible]->130.153.15.92)
6:MS-SQL Worm propagation attempt(61.159.15.2->130.153.107.207)
7:ICMP PING NMAP(221.190.226.96->130.153.158.77)
8:MS-SQL Worm propagation attempt(210.211.184.58->130.153.241.79)
9:ICMP Destination Unreachable (Communication Administratively Prohibited)(62.219.135.48->130.153.153.19)
```

Figure 2.6: IPMatrix [38] uses two coordinated scatterplots, one for the A and B "internet level" address spaces and one for the C and D "local level" address spaces. Each one displays detected attacks by address spaces as pixels colour coded to the type of attack. These points are superimposed over a grid of square bins showing the number of attacks by block of addresses. Histograms along each axis display total activity for each value.

Heat maps are a special variety of scatterplot, designed to map the same two dimensional correlations with an extra dimension encoded to the mark value variable. For heat maps, this extra dimension can be time, which helps to represent the density correlation patterns through time. When properly configured these types of visualisations help to characterise activity over a period of time and provide historical visual profiles for variable correlation. PortVis [49] uses them to display port activity over time, but this approach has also been used for objectives other than monitoring to characterise traffic and recognise protocols in encrypted traffic [44], [79] using packet sizes (Fig 2.7).

While scatterplots are limited to representing events punctually or with any distinction in time, heat maps can help operators understand the historical evolution of a pattern, and use that context to understand more recent events. For example, the activity on a host for the past hour can be compared with a heat map describing traffic over the last day, or even for the entire lifetime of a host. However, while this approach solves issues related to time, designers of monitoring tools are still effectively limited to three dimensions per representation.



(a) HTTP

(b) SMTP

(c) AIM

(d) SSH

Figure 2.7: In [44], [79], traffic is characterised using heat maps of packet bigrams. Point coordinates are mapped to the size of the first packet and second packets and value is mapped to the quantity of that bigram. These visual motifs display four different protocols with characteristic and rapidly recognisable patterns thanks to the distinctive relationship between the sizes of sent and received packets in different protocols.

When multiple numeric dimensions need to be correlated, one approach is to split these into multiple scatterplots highly coupled either by shared or relatable dimensions, and arranged to encourage visual comparison. Discussed previously, IPMatrix [38] splits IP addresses into two and displays scatterplots for internet level and local level traffic. Linked visualisations in PortVis [49] complement each other to provide some level of interactivity.

Using multiple visualisations which share a common dimension, such as time, or sharing filters, operators can use dashboards to simultaneously follow changes in multiple synchronised datasources. The number of available visual variables is a limit for single visualisations, which makes this approach more scalable. However, because adding extra views requires more physical display space, the full interface is quickly limited by available space. Also, due to the distance between each visualisation and the isolation of each component, seeing relations between dimensions and detecting correlations is more difficult than when these dimensions share space.

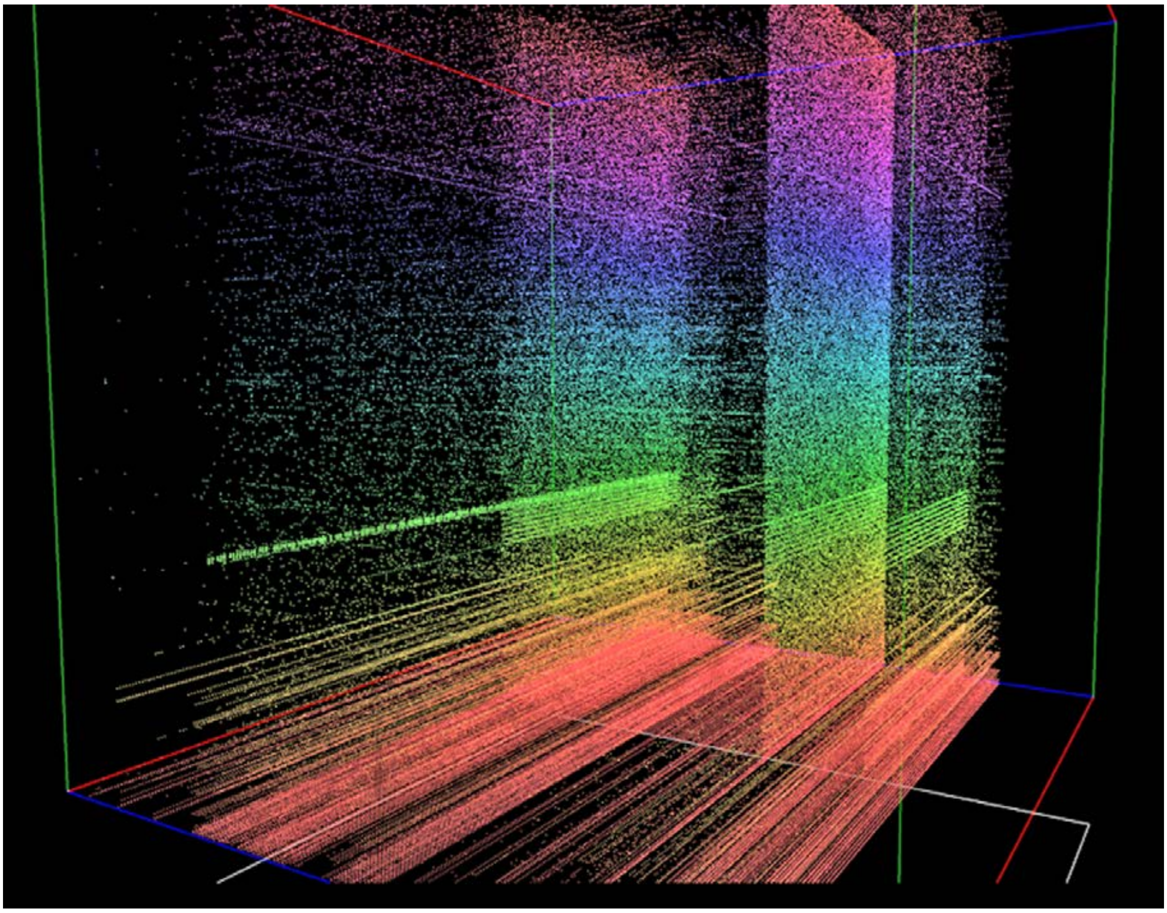


Figure 2.8: [36] visualises Snort and Bro scan detection using 3D scatterplots. Connections are mapped by source address, destination address and destination port. These fade out with time to help highlight the most recent ones. Using this configuration, millions of connections can be represented in near real-time, allowing security practitioners to visually detect scan patterns over time.

Instead of separating data into multiple views and sacrificing direct pattern recognition, some use three dimensional visualisation. For instance, 3D scatterplots can simultaneously map source IP address, destination IP address and destination port in one single space have been used in [36]. This allows operators to directly see correlations between all three dimensions at once, with the additional risks of elements occluding others. Perspective also makes comparing values difficult. Additionally patterns require exact viewpoint orientation, panning and zoom to be seen (Fig 2.8).

Using three dimensional visualisation quickly solves some correlation issues. By mapping ordinal dimensions to the position of each mark, and two further dimensions to the colour and value (opacity) visual variables, a single visualisation can help correlate five dimensions. However, five dimensions is more or less a hard limit[5], and three dimensional visualisation suffers perception issues: perspective can make reading values misleading, occlusion means that marks can always be hiding others, and some patterns might only be noticed from certain angles, leading to a continuous need for interaction.

⋈

To monitor security event data for familiar patterns, security administrators need visualisations which can display these patterns without needing extra configuration. Scatterplots and heat maps can reliably display patterns for a small number of dimensions. However, to correlate beyond this limit, practitioners will need to split events over multiple visualisations.

In the next subsection, we discuss how security tools help practitioners keep in touch with the current situation.

2.1.2 Situational awareness

Changes in visual patterns are signs of potential anomalies. To react effectively the situation needs to be understood quickly. Proper context and readily available inspection are essential.

Decision-making is therefore directly dependant on situational awareness and implies being able to quickly obtain answers to basic but essential questions. VisAlert [21], [46], [47] proposes a visualisation designed specifically for understanding the nature, time and location of events thanks in part to context for each of these factors to compare an event with its history and location (Fig 2.9).

More generally, composite visual representations combine multiple types of marks in close proximity to help users relate different dimensions of a dataset. They have the advantage of being tailor made for specific situational awareness, and provide answers quickly with rich context. However, this approach also restricts extendability to other questions and different sets of dimensions. Bespoke design also requires a period of adaptation to unfamiliar visualisation.

In contrast with a unified single central visualisation, Overflow [23] distributes data dimensions across an interface with multiple adjacent visualisations (Fig 2.10).

[5] Evolution has favoured rapid counting of predators or prey in small groups, but this talent was less useful when facing large groups. This resulted in almost pre-attentive speeds when counting small groups of marks, which do not scale very far.

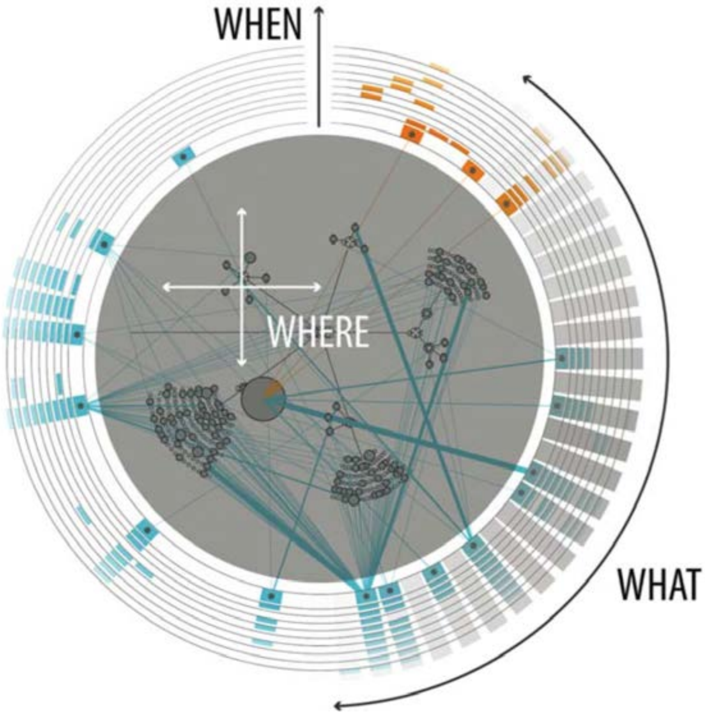
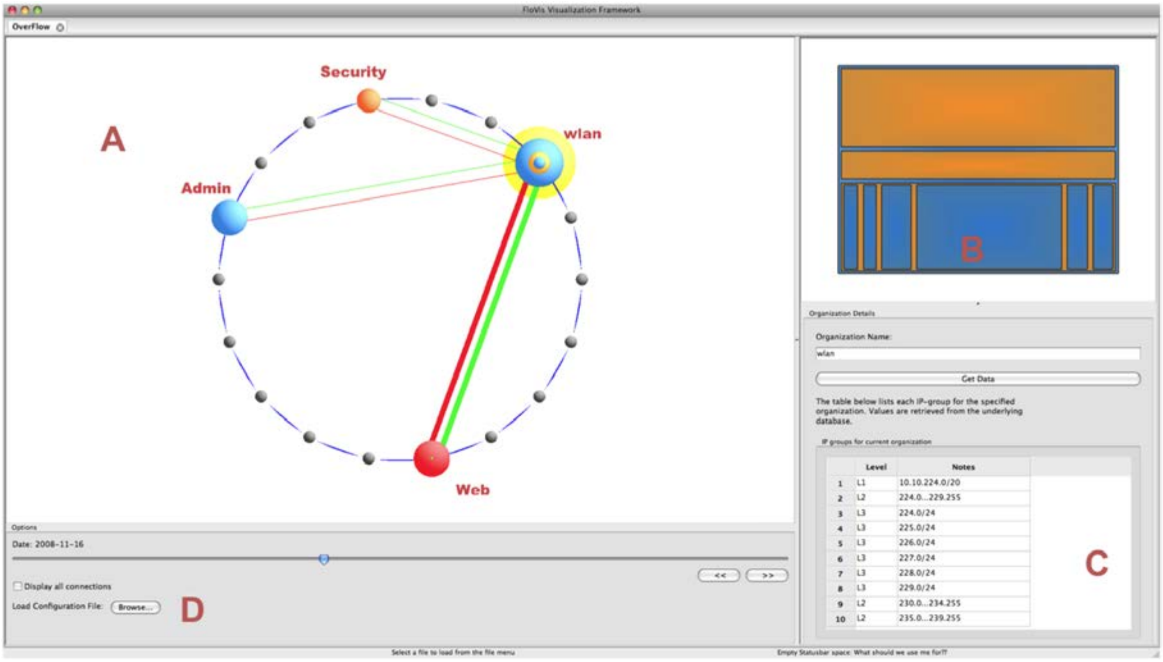


Figure 2.9: In VisAlert [21], [46], [47], alerts are mapped to a coloured radial slice of a surrounding ring corresponding to its type, and moves outwards as it ages. To show where the alert took place, the space inside the ring houses a spacial or organisational visualisation to which each alert is linked. When used for network intrusions for example, this central visualisation can be a graph representation of the network.

Using simplified and more familiar views, operators understand new information with less adaptation and can more easily configure which dimensions to display. Following changes in more dimensions and datasets simultaneously is possible when these are synchronised and arranged in an efficient layout. However, finding an efficient layout is a challenging task. With the wrong choice, following and relating dimensions visualised in views far from each other becomes a difficult cognitive task, and using simplified representations removes meaning sometimes crucial to good background context.

NIMBLE [56] employs a more concrete design approach by organising monitored hosts into graphs of cards and directly displaying connections. Explanations for events and recommendations are then directly written in the card and links (Fig 2.11).

Concrete representations such as node-link graphs communicate the semantics of the data well. Following the cognitive dimensions *closeness of mapping* and *hard mental operations* (page 19), node-link graphs help people understand data as a network of connections. Using these types of visual metaphors, operators can more easily use their experience and work in familiar territory. However, these ap-



Date: 2008-11-16

□ Display all connections

Load Configuration File:

Organization Details

Organization Name: wlan

The table below lists each IP-group for the specified organization. Values are retrieved from the underlying database.

IP groups for current organization

	Level	Notes
1	L1	10.10.224.0/20
2	L2	224.0...229.255
3	L3	224.0/24
4	L3	225.0/24
5	L3	226.0/24
6	L3	227.0/24
7	L3	228.0/24
8	L3	229.0/24
9	L2	230.0...234.255
10	L2	235.0...239.255

Select a file to load from the file menu

Empty Statusbar space: What should we use me for??

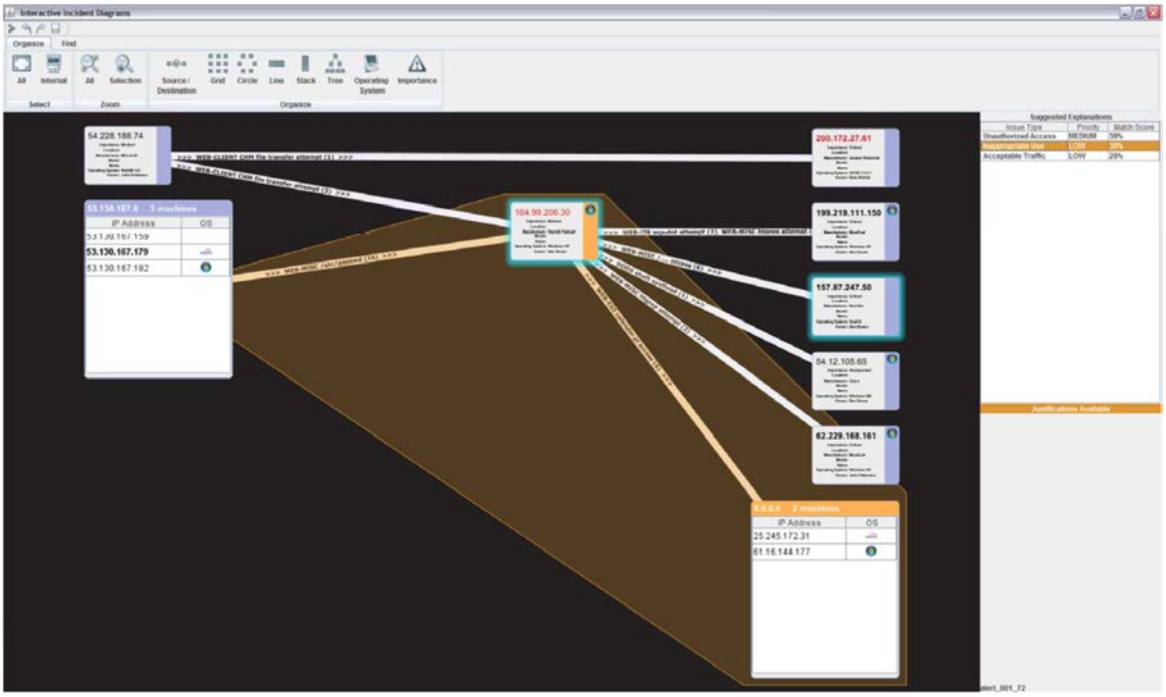
Figure 2.10: OverFlow [23] provides three views. The first is a radial visualisation displaying the different network elements. Rings inside each element show a simplified representation of its hierarchy, and lines between the network elements show ingoing and outgoing communication between them. The second visualisation displays the detailed hierarchy for a selected element using a tree map, with the same colours as the simplified ring display. The last visualisations show the ip-groups for network elements.

proaches can rapidly become complex when representing large scale datasets, and can be misleading when the layout process is automatic.

Force based layouts lead people to see patterns which do not come from the original data. In these visualisations the position variable for each mark is the result of repulsion and attraction forces seeking equilibrium. Any emerging patterns are the result of this progressive physical simulation. They are by definition difficult to predict or reproduce and can show radically different results depending on initial placements or additional data.

⋈

To maximise the situational awareness of security practitioners, security visualisation tools are designed to answer fundamental questions about the type of situation which could be encountered. Using visualisations which provide spatial and temporal context using visual analogies close to the problem world keeps practitioners in familiar territory and helps them understand any situational changes faster.



IP Address	OS
53.130.167.159	
53.130.167.179	
53.130.167.182	

IP Address	OS
25.245.172.31	
61.16.144.177	

Suggested Explanations

Issue Type	Priority	Match Score
Unauthorized Access	MEDIUM	58%
Inappropriate Use	LOW	30%
Acceptable Traffic	LOW	29%

Figure 2.11: NIMBLE displays a graph-based visualisation which links information cards based on triggered alerts. Each card represents one or more hosts with more or less details about them. The links then carry text descriptions for the type of alert concerning the connected nodes.

In the next subsection, we discuss how security visualisation tools can stay useful when faced with large quantities data.

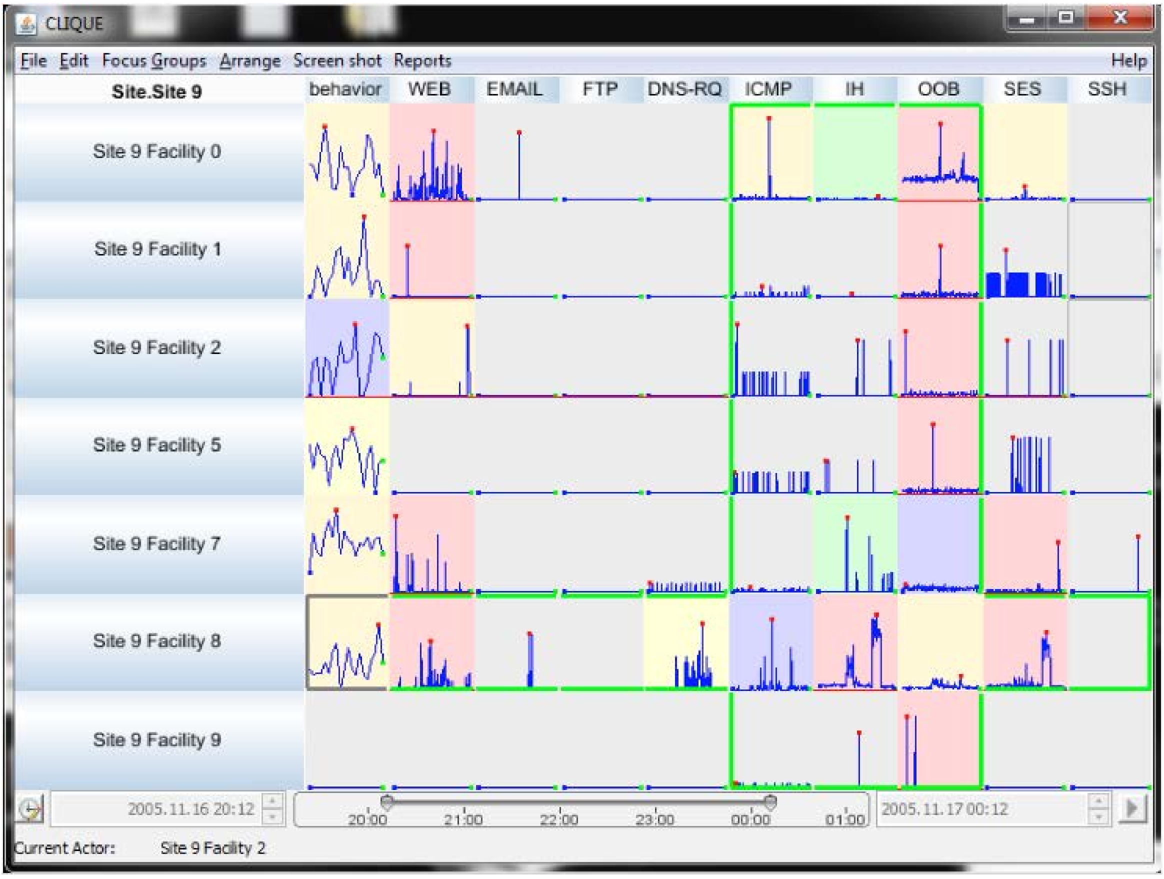
2.1.3 Responding to scale

Monitoring services need to manage large numbers of datasources from entire networks of hosts. For instance, data centres contain many nodes running hundreds of services all connected by complex networks. Monitoring this data with visualisation means managing the same increasing scales of data, often with requirements for real-time updates and data history.

The physical constraints of displays usually limit real-time visualisation to filtered datasets. To improve the visualisation of real-time network data, Daniel *et al.* [14] describe two visualisations to help alleviate these problems which make heavy use of sparkline type visualisations (Fig 2.12).

The term *sparkline* was introduced in [72] to designate "small, high resolution graphics embedded in a context of words, numbers, images". These small and compact charts are intended for communicating large amounts of data in small spaces, and can often be used in the same space as words in text or tables for example.

Small multiples are series of graphics sharing the same design and configuration of variables to display the changes of an index variable. This unshifting design makes changes in information clearly perceptible as differences in the state "snapshots". When aligned according to the index variable, trends can be made clear.



2005.11.16 20:12

2005.11.17 00:12

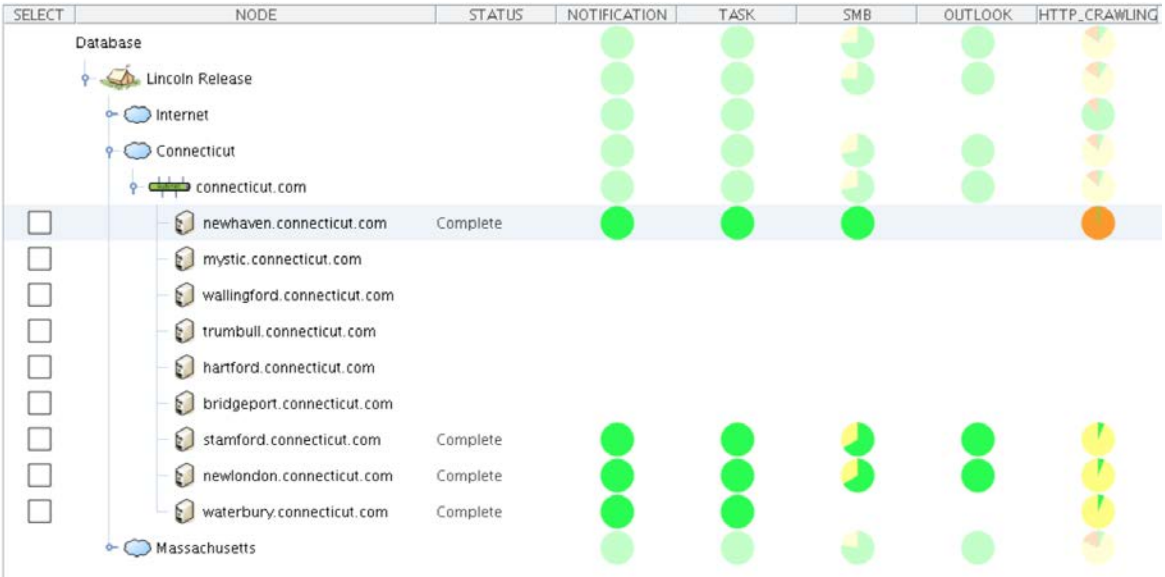
Current Actor: Site 9 Facility 2

Figure 2.12: Based on LiveRac[6], CLIQUE [14] offers a tabular organisation of data with columns for each service and rows for users. Each cell initially contains a sparkline type visualisation which can be expanded to show more detail.

Yu *et al.* [80] use testbeds for visualising services and machines in grids of miniature pie charts. Colour coded warnings and errors stand out and problems can quickly be identified (Fig 2.13). Clockview [37] was designed for representing the state of hundreds of hosts on a network. Once again, small charts represent hosts which are then superimposed with a graph representation of the network. Each chart shows the evolution of activity on a host over time and can be tuned for the desired granularity of time (Fig 2.14). Pearlman *et al.* [54] choose to represent hosts as link-node graph to explicitely represent the network organisation. Each host is once again represented using a compound circular visualisation composed of rings. Each

[6] Liverac is a visualisation tool used to explore data with numerous dimensions from large numbers of hosts.

ring summarises the running services for a given period of time and proportionally to the used resources (Fig 2.15).



SELECT	NODE	STATUS	NOTIFICATION	TASK	SMB	OUTLOOK	HTTP_CRAWLING
	Database		●	●	●	●	●
	Lincoln Release		●	●	●	●	●
	Internet		●	●			●
	Connecticut		●	●	●	●	●
	connecticut.com		●	●	●	●	●
□	newhaven.connecticut.com	Complete	●	●	●		●
□	mystic.connecticut.com						
□	wallingford.connecticut.com						
□	trumbull.connecticut.com						
□	hartford.connecticut.com						
□	bridgeport.connecticut.com						
□	stamford.connecticut.com	Complete	●	●	●	●	●
□	newlondon.connecticut.com	Complete	●	●	●	●	●
□	waterbury.connecticut.com	Complete	●	●			●
	Massachusetts		●	●	●	●	●

Figure 2.13: Yu *et al.* [80] propose a testbed visualisation tool with trees for displaying hierarchy, graph and matrix views for displaying network flows, timelines for displaying user activity, but also a device status visualisation, using small multiple grids of pie chart glyphs. Each row corresponds to a specific host, and each column a service. Each glyph displays current status and current alert proportions, and provides details on demand.

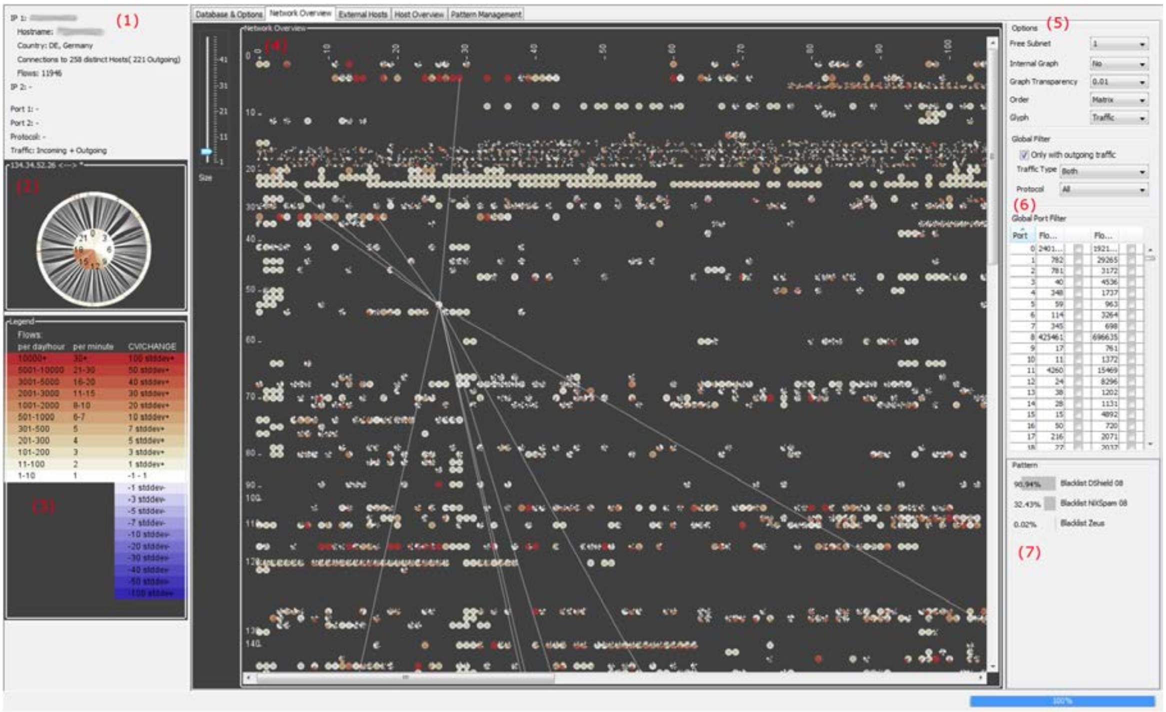
DAEDALUS [35] uses 3D visualisation to monitor large numbers of connections and events on multiple organisations with known address spaces and "darknet" spaces (Fig 2.16).

Using three dimensions extends the limits of screen space to encompass depth. This adds more usable space for visualisations, but comes with previously discussed disadvantages. While the objective of scaling a visualisation is to represent more information, problems with occlusion and perspective in 3D visualisations still limit data density to the available physical pixels. To alleviate these issues, use of animation and visual hints is essential.

⋈

Monitoring tools make significant use of correlation visualisations with fixed configurations and often combine visualisation techniques in different views to cover as much data as possible:

- combinations of pre-configured visualisations for pattern recognition, such as scatterplots and parallel plots;
- visualisation adapted to parallelisation (pixels, radial charts. . .);



IP 1: [redacted]
Hostname: [redacted]
Country: DE, Germany
Connections to 258 distinct Hosts(221 Outgoing)
Flows: 11946
IP 2: -

Port 1: -
Port 2: -
Protocol: -
Traffic: Incoming + Outgoing

Legend
Flows:

per day/hour	per minute	CVICHANGE
10000+	30+	100 stddev+
5001-10000	21-30	50 stddev+
3001-5000	16-20	40 stddev+
2001-3000	11-15	30 stddev+
1001-2000	8-10	20 stddev+
501-1000	6-7	10 stddev+
301-500	5	7 stddev+
201-300	4	5 stddev+
101-200	3	3 stddev+
11-100	2	1 stddev+
1-10	1	-1 - 1
		-1 stddev-
		-3 stddev-
		-5 stddev-
		-7 stddev-
		-10 stddev-
		-20 stddev-
		-30 stddev-
		-40 stddev-
		-50 stddev-
		-100 stddev-

Options
Free Subnet: 1
Internal Graph: No
Graph Transparency: 0.01
Order: Matrix
Glyph: Traffic

Global Filter
☑ Only with outgoing traffic
Traffic Type: Both
Protocol: All

Global Port Filter

Port	Flo...		Flo...	
0	2401...	□	1921...	□
1	782	□	29265	□
2	781	□	3172	□
3	40	□	4536	□
4	348	□	1737	□
5	59	□	963	□
6	114	□	3264	□
7	345	□	698	□
8	425461	□	696635	□
9	17	□	761	□
10	11	□	1372	□
11	4260	□	15469	□
12	24	□	8296	□
13	38	□	1202	□
14	28	□	1131	□
15	15	□	4892	□
16	50	□	720	□
17	216	□	2071	□
18	22	□	2037	□

Pattern
90.94% Blacklist DShield 08
32.43% Blacklist NiXSpam 08
0.02% Blacklist Zeus

Figure 2.14: In ClockView [37] each host is represented as a radial chart, similar to a clock, again displaying activity for 24 hours, but as a glyph, and easily comparable. For network structure awareness, a communications graph can be superimposed on top of the view. After interesting hosts have been found, focusing on them replaces the global view by other better suited visualisations. Activity is better shown using a pixel matrix with better granularity for over time, and the individual clock charts are shown in a parallel plot configuration for correlation. An even more detailed view displays a port matrix for examining the interactions between two specific machines.

- motion and transitioned updates to show change in time.

These shared factors are well suited to helping the monitoring persona keep track of events. They help maximise the quantities of displayed information while keeping interaction to a minimum so the operator can focus on recognising patterns.

They also share several limits due to their specialisation: they are often designed for handling a single type of data using a fixed visualisation to help operators to see patterns. Following this approach implies that only known issues related to these data sources will be efficiently caught. This seems typical of monitoring visualisation, which need to provide a familiar view to be effective and help maintain a first line of defence. This characteristic explains the need for analysis tools.

⋈

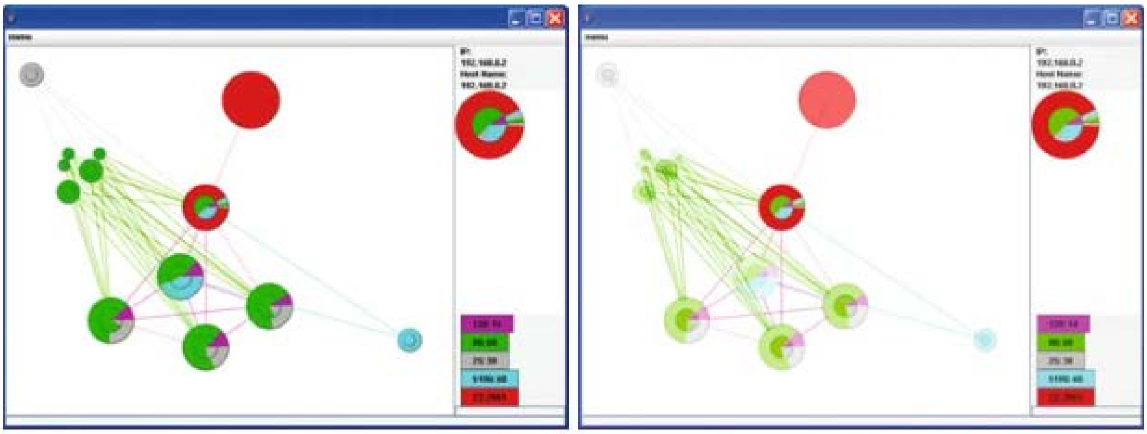


Figure 2.15: Pearlman [54] *et al.* use compound glyphs to display running services. Here each node represents its currently hosted services as a hierarchical donut chart. These services are displayed proportionally, with outer rings representing their recent state, enabling people to see a partial short-term history. Simple hosts have simple representations, and can be rendered smaller to make important nodes more prominent.

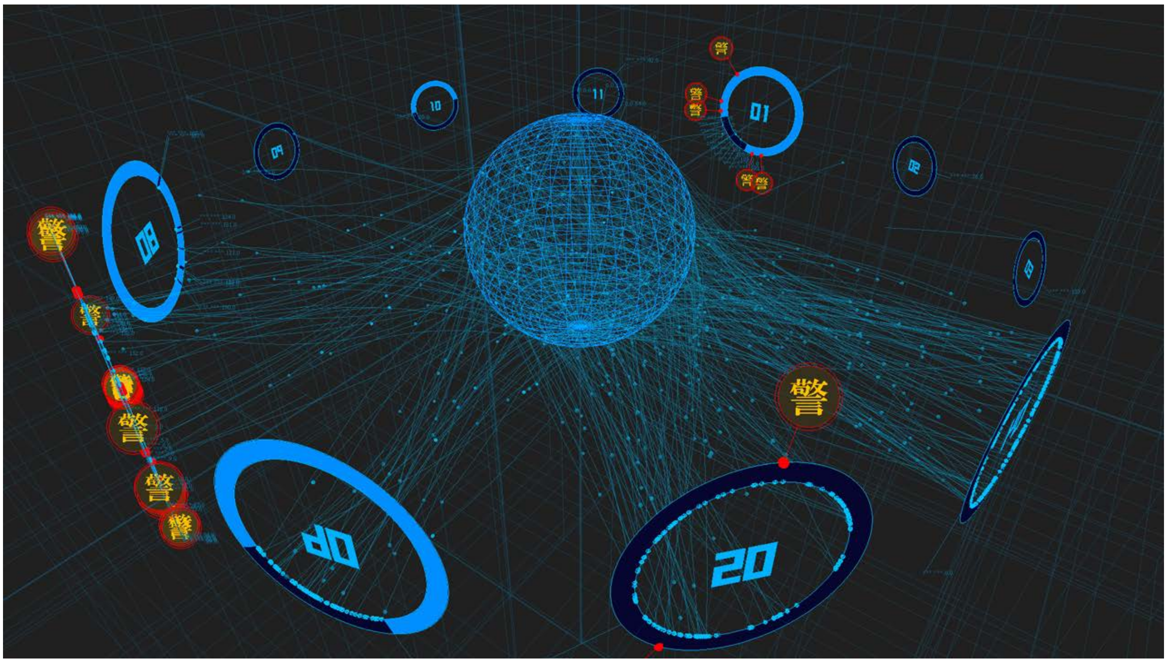


Figure 2.16: DAEDALUS [35] is a uses three dimensional visualisation. The centre of the view is occupied by a sphere divided into 256 by 256 points and capable of representing mapping an entire /16 network. From this sphere, emanating lines represent connections and points travelling along them are events. The sphere is surrounded by ring charts which each represent a monitored organisation with occupied address spaces and dark net spaces. Alarms triggered by these events are made visible using the Chinese character meaning "caution".

Monitoring tools tools allow operators to watch for familiar patterns in large quantities of data, and are built with realtime situational awareness in mind:

- simple correlation visualisations such as scatterplots and heat maps;
- designs which help with situational awareness by answering fundamental questions and providing relatable visual analogies;
- visualisations which can be scaled down and highly parallelised.

Most of these tools are built to monitor specific situations. Their designs are largely static with very few options for configuration and often for specific sources of information. This can make them very efficient, but also reduces adaptability.

The next section discusses tools with very different objects, those designed for security data analysis.

2.2 VISUALISATION FOR ANALYSIS

Security data analysis follows the tasks of monitoring when intrusions are detected or when new anomalies cannot be explained. Operators use analysis tools to better understand situations and the process of events leading to them.

Analysing data with visualisation implies exploring different configurations and types of visual representations in search of meaningful information. The analysis is guided either by a specific objective, such as finding a flow of intrusions, or aims to find interesting patterns and signs of potentially malicious activity without critical time constraints. To this end, tools for visual analysis are designed for data exploration and often for specific data sources, but still need to be flexible.

Visualisation tools for security analysis are designed to accomplish two goals: provide a method of exploration through search cycles and help reconstruct narratives.

2.2.1 Search processes

In order to provide a forensic search experience and better explore security data, tools are designed to encourage searching using interaction cycles. These function by reacting to operator feedback and updating the visualisations or visualised data. As time is less a factor than when monitoring systems, operators are free to inspect details and backtrack to previous steps.

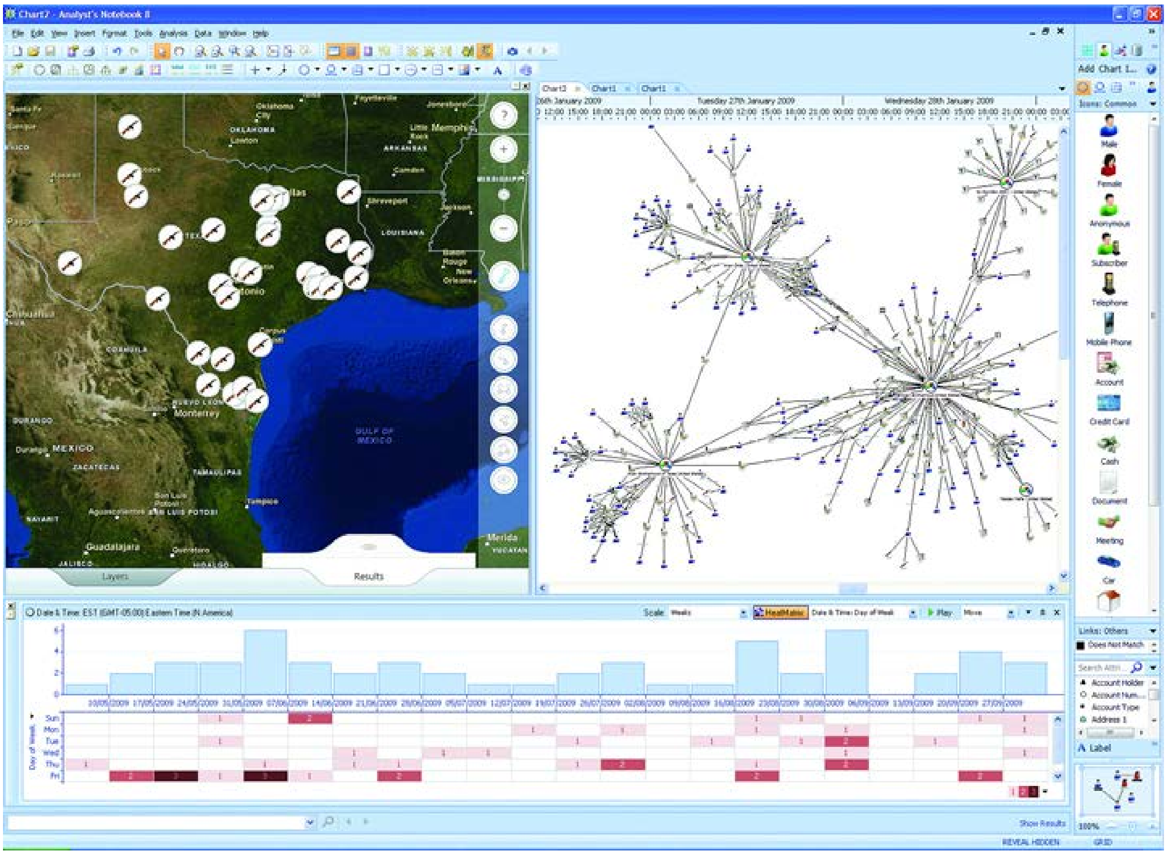


Figure 2.17: The Analyst's Notebook [3] displays data using maps to locate forensic data, node graph to visualise links between them as well as matrices, bar charts and tables for details and distributions.

Commercial tools such as The Analyst's Notebook [3] provide visualisations adapted for exploring information to find links in real-world forensic data (Fig. 2.17). Using these tools the operator can cycle between the visualisations using each as a filter for the others to refine the search until only the essential information is found.

NVisionIP [40] provides three levels of detail in separate views for exploring NetFlow data to offer a drill down approach typical in visualisation tools (Fig 2.18). Instead of using sequential visualisations to provide drill-down exploration, Harrison *et al.* [27] use multiple simultaneous synchronised views. Each view reacts to changes in the others and can filter the shared information. This incremental filtering gradually helps the operator focus on points of interest (Fig 2.19). BGP Eye [55] was designed to help visualise and explore border gateway protocol (BGP) anomalies that are signs of potential malicious activity. Multiple successive views offer different complementary points of view of the anomalies (Fig 2.20).

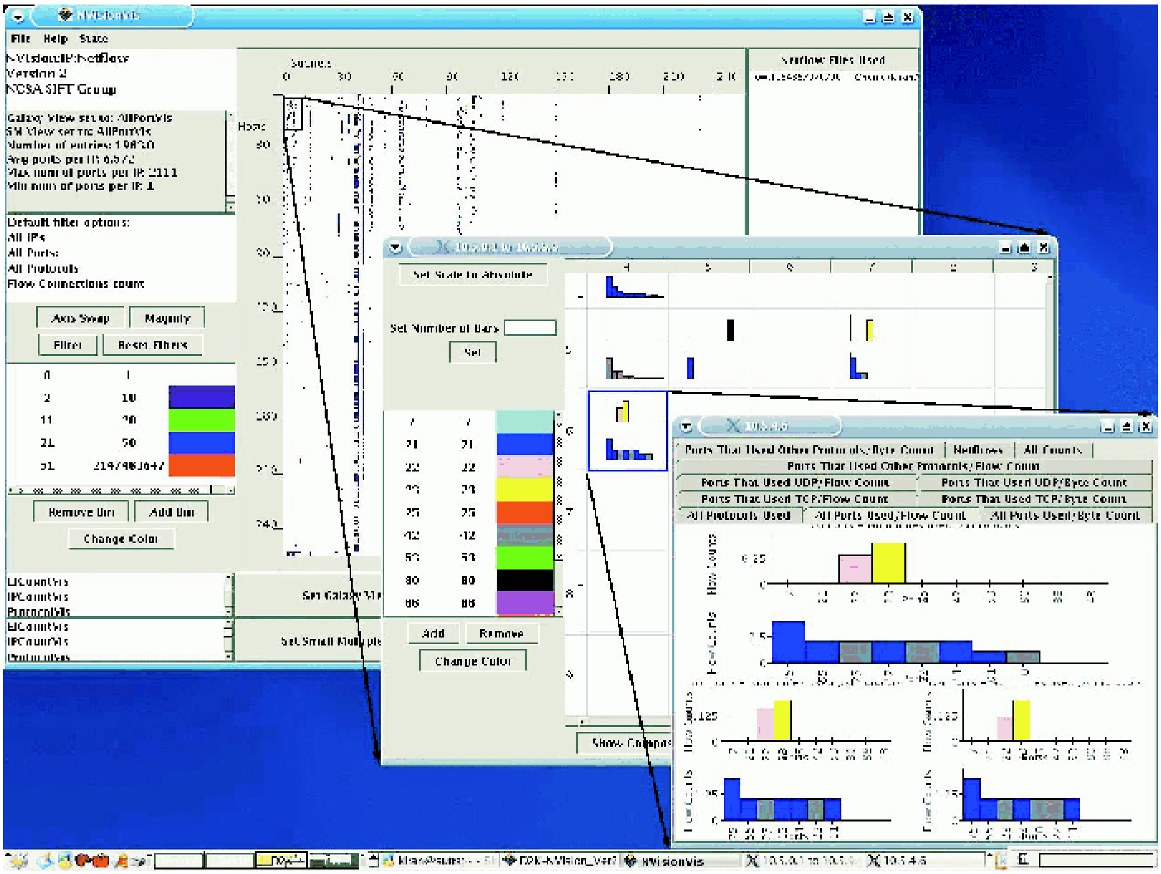
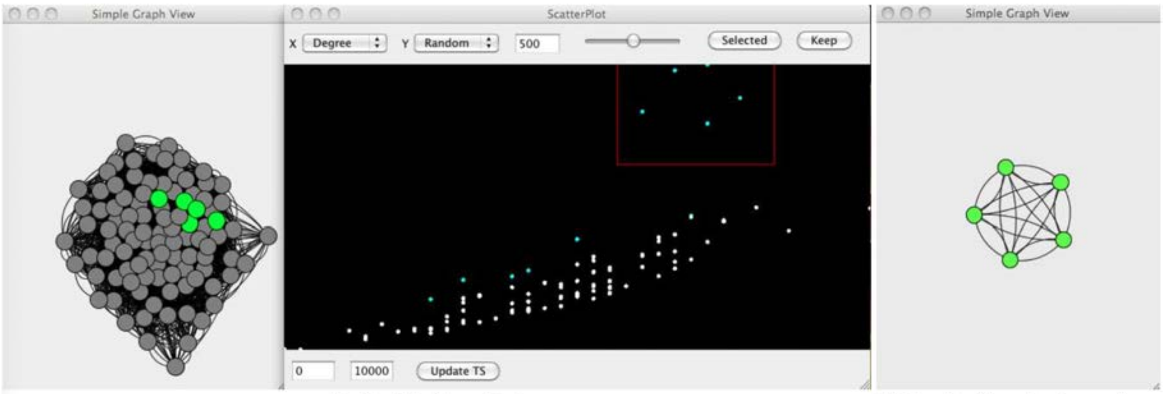


Figure 2.18: The three main views in NVisionIP [40]: the main view displays flows by subnet and host using a "galaxy view" scatter plot. Selecting in this view opens a small multiple view to compare hosts, which can then also open a view for inspecting machine details. Flows can be marked with colours to help follow them in each view.

Each of these approaches uses multiple views to show different parts of a dataset. In each case these views are tightly related and linked together to form either a hierarchy or a multi facet interface. Interaction with any single view through selection, panning or zooming translates to filters for the dataset as a whole. When the dataset is filtered, the exploration progresses to a new data subset. This triggers updates or replacements of current representations depending on the nature of this subset.

In a hierarchy of views, each new subset of information is represented using a replacement configuration of views. In a multi facet interface, the effects of these actions are displayed through updates to the concerned views. These results help operators analyse security information using a consistent drill-down type exploration process.

A hierarchy based approach adopts a structure close to the drill-down metaphor. When the analyst focuses on a different data subset, new views are generated which can each be designed for the aspects of the new data. However, exchanging one view for another means



(a) Graph before selection

(b) Graph after selection and update

Figure 2.19: Harrison *et al.* propose [27] multiple synchronised views for spatial analysis. The first view offers a graph based layout of the network. For temporal analysis, a time histogram shows all events for a given period. The third view shows the updated graph after time filtering has been applied using the time histogram.

losing perspective on other parts of the data which can provide context critical to understanding the nature of the new dataset. An approach using faceted views conserves the same multiple views of the dataset throughout the exploration process. Although this means that representations for new information are constrained, this also helps operators follow the links and relations between different views. Is is worth noting that in both cases, the exploration history is hidden or even lost. This makes returning to previous steps a difficult task, which might require taking manual notes.

2.2.2 Scene reconstruction

Cyclical search processes make interactions with analysis tools closer to a the search methodology proposed by Shneiderman:

- using a global view to search for anomalies,
- removing groups of items which seem uninteresting,
- and inspecting remaining areas in detail to find relatable points within them.

Operators search for outliers then link data together to reconstruct a narrative describing the events and their protagonists. For operators to follow the designed search process, they need to see meaningful links between data which could help reconstruct events. Interfaces are designed to juxtapose different but related views of datasets and provide intuitive means to link information from one view to another.

The Network Visualiser (TNV) [25] was designed as an alternative to Wireshark, augmented with visualisation to provide a global

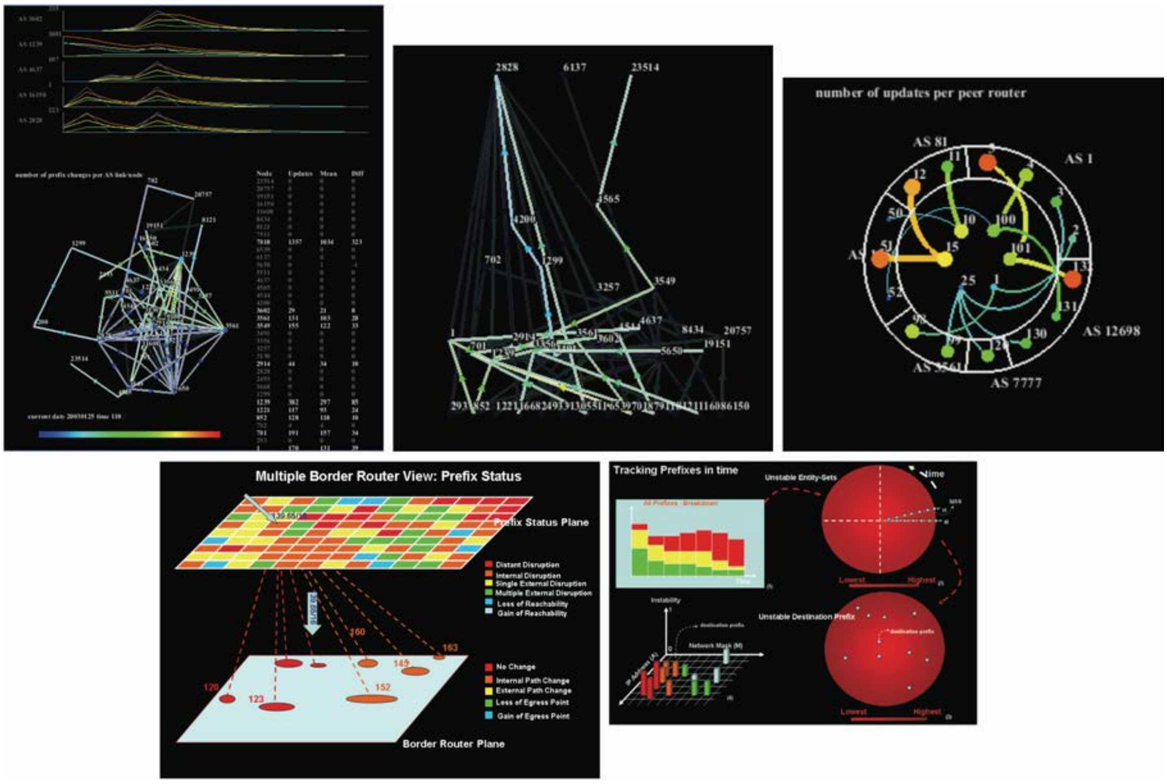
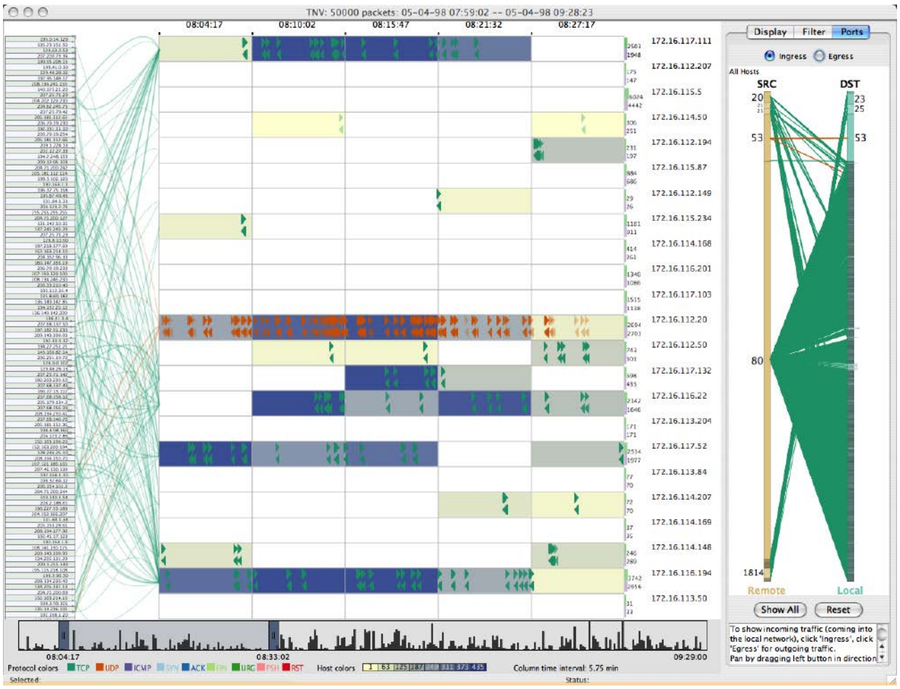


Figure 2.20: BGP Eye [55] uses three visualisations: The first is a force-based graph layout showing BGP events triggered by specific Autonomous Systems (AS). The second uses a path distance based layout with source nodes at the bottom, sink nodes at the top and all intermediate nodes placed between according to a distance metric determined using a breadth-first-search algorithm. The third uses a circular representation to display connections between observation routers in an inner ring and peer routers in an outer ring. The last view is a complex association of two plane visualisations, one showing concerned border routers, and the corresponding prefix statuses, enabling security practitioners to see which prefix/router has recently seen changes to routing paths.

view of network packets. First, operators can quickly obtain a global understanding of packet capture files. The interface lays out traffic from left to right, with sources on the left and destinations on the right. An accompanying histogram provides an interactive timeline (Fig 2.21). Operators follow traffic from source to destination, while passing through a matrix which describes the nature of the information. Using the same left to right analogy for flow of information from clients to servers, Portall [19] helps the user understand client and server processes and how they communicate with each other (Fig 2.22).

Tools such as PicViz [71] and SeeSoft [16] were designed to handle numerous types of log files and concentrate on a single highly configurable visualisation. PicViz centres its interface around a parallel plot to help detect correlations and points of connection between dimensions (Fig. 2.23), while SeeSoft provides a textual representation



To show incoming traffic (coming into the local network), click 'Ingress', click 'Egress' for outgoing traffic. Pan by dragging left button in direction

Column time interval: 5.75 min

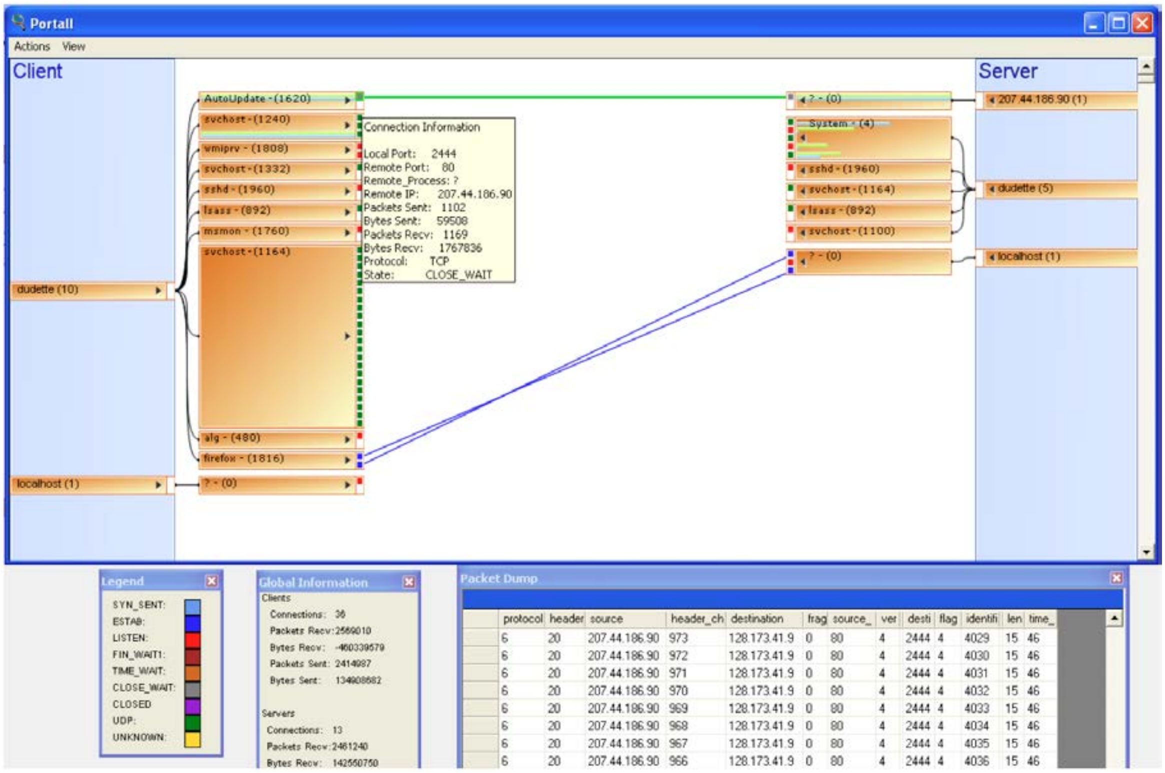
Figure 2.21: TNV [25] is designed around a central matrix visualisation displays circulating packets, annotated by direction and colour coded by protocol type. On either side, addresses are listed and linked to the different packet flows. On the right, a parallel plot shows port usage, and underneath, a histogram displays traffic evolution and enables filtering. As such, filters surround a horizontal progression of packets, making it easier to comprehend network traffic.

of log files with a colour code for differentiating and grouping events (Fig. 2.24).

BURN (Baring Unknown Rogue Networks) [60] helps with the exploration of autonomous systems by malicious activity. To visualise datasets of autonomous systems (AS) containing rogue activity, Roveta *et al.* use animation to make malicious activity stand out more and transitions to display the evolution of the concerned systems (Fig 2.25). The different views allow operators to see data by rank of maliciousness, by geographic location and by historical activity. By moving between these aspects by following points of interest, operators can reason about the different AS and build a richer picture of events.

⋈

Visualisation tools designed for analysis help operators to discover links between security data. With visualisations sharing dimensions and a layout to encourage comparison of these visualisations, operators can reason about the differences between these properties and actors, but also the similarities. This allows them to form groups of



Connection Information

Local Port: 2444
Remote Port: 80
Remote_Process: ?
Remote IP: 207.44.186.90
Packets Sent: 1102
Bytes Sent: 59508
Packets Recv: 1169
Bytes Recv: 1767836
Protocol: TCP
State: CLOSE_WAIT

Legend

SYN_SENT:	🟦
ESTAB:	🟦
LISTEN:	🟥
FIN_WAIT1:	🟥
TIME_WAIT:	🟧
CLOSE_WAIT:	⬜
CLOSED	🟪
UDP:	🟩
UNKNOWN:	🟨

Global Information

Clients
Connections: 36
Packets Recv: 2569010
Bytes Recv: -460339579
Packets Sent: 2414987
Bytes Sent: 134908682

Servers
Connections: 13
Packets Recv: 2461240
Bytes Recv: 142550750

Packet Dump

protocol	header	source	header_ch	destination	frag	source_	ver	desti	flag	identifi	len	time_
6	20	207.44.186.90	973	128.173.41.9	0	80	4	2444	4	4029	15	46
6	20	207.44.186.90	972	128.173.41.9	0	80	4	2444	4	4030	15	46
6	20	207.44.186.90	971	128.173.41.9	0	80	4	2444	4	4031	15	46
6	20	207.44.186.90	970	128.173.41.9	0	80	4	2444	4	4032	15	46
6	20	207.44.186.90	969	128.173.41.9	0	80	4	2444	4	4033	15	46
6	20	207.44.186.90	968	128.173.41.9	0	80	4	2444	4	4034	15	46
6	20	207.44.186.90	967	128.173.41.9	0	80	4	2444	4	4035	15	46
6	20	207.44.186.90	966	128.173.41.9	0	80	4	2444	4	4036	15	46

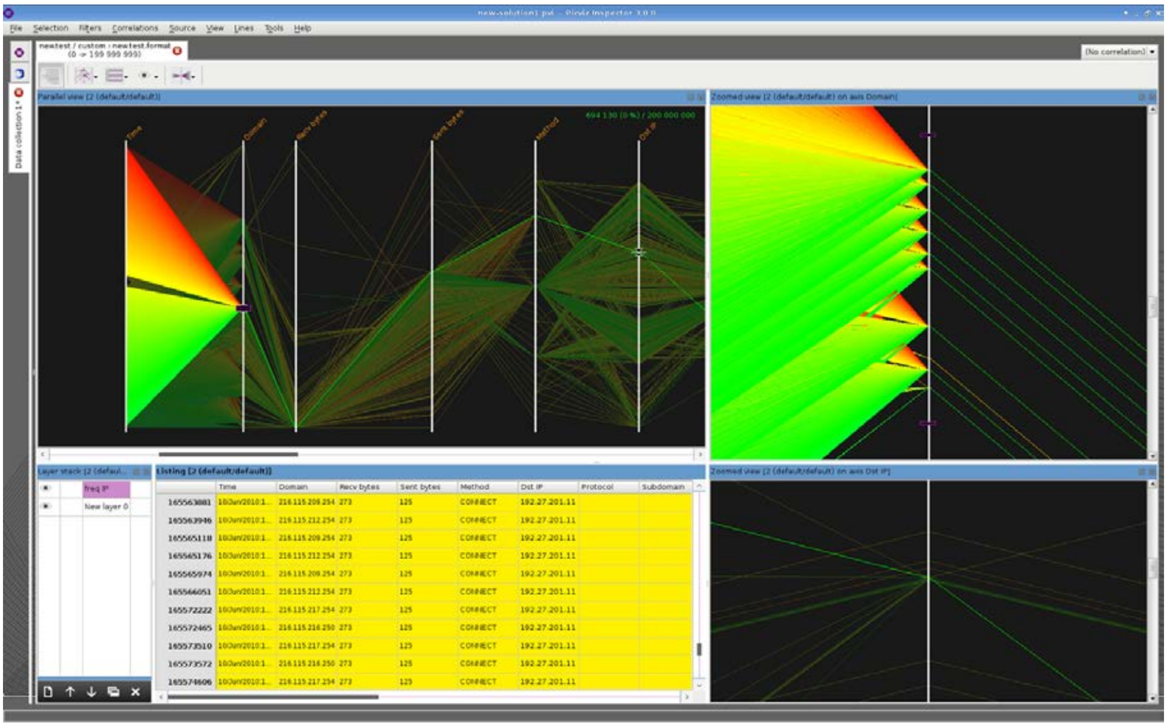
Figure 2.22: In Portall [19] clients are listed on the left and servers on the right, and their running processes are linked in a graph, displayed as boxes with process details, notably connection activity histogram.

events, move from data to data, and through the links which emerge, try to reconstruct scenarios.

To summarise, in contrast with monitoring tools, analysis tools encourage taking time for exploration, and allow users to focus on parts of the data. Again, multiple views are often available using the same correlation visualisations but these are modifiable, with more possibilities for interaction. They are built to allow for search cycles with the objective of understanding and recreating events:

- linked combinations of visualisations for pattern and trend recognition;
- strong use of filtering to configure visualisation parameters;
- motion used as true variables to indicate differences and to mark importance.

Many of these tools are designed to help explore a single type of data, such as Snort alerts or pcap files. Analysing issues with systems often requires relating multiple sources of data, such as proxy access logs and authentication logs, to properly reconstruct events. This implies switching between tools during an exploration session

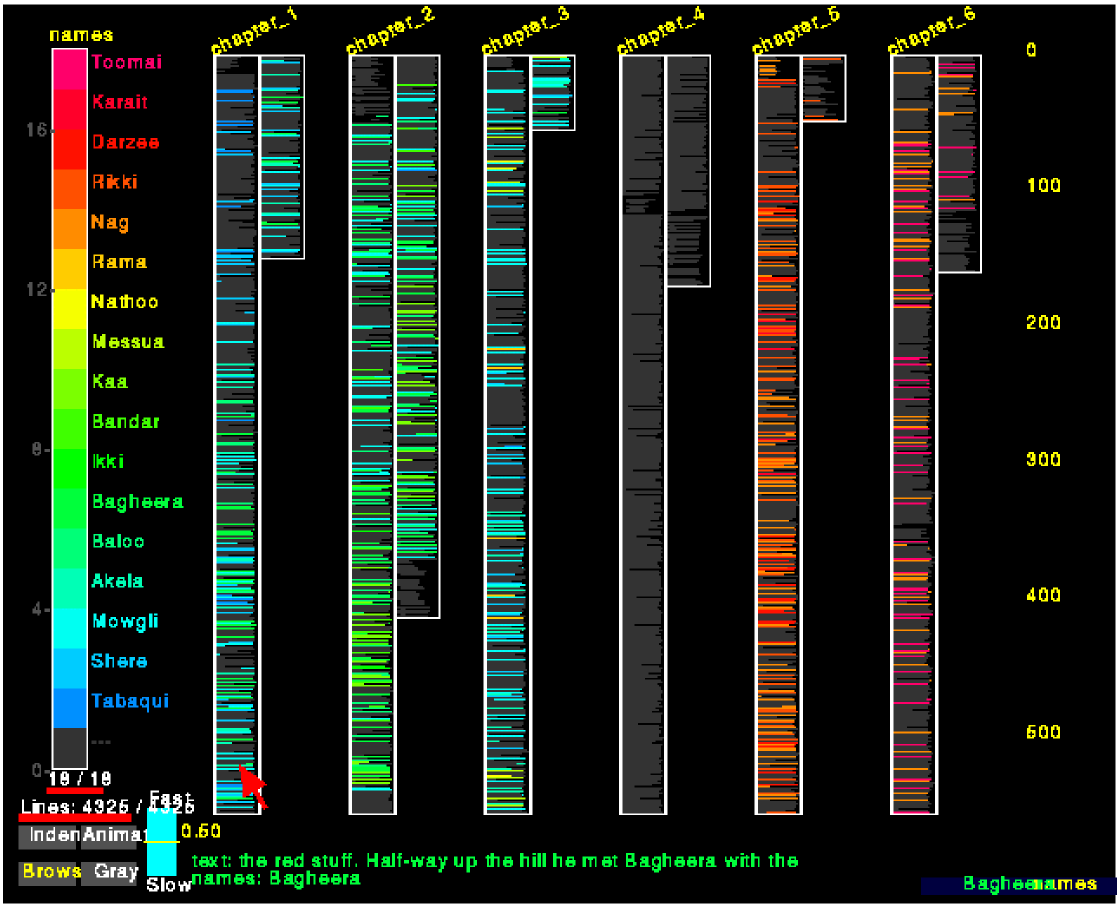

File Selection Filters Correlations Source View Lines Tools Help
newtest / custom : newtest.format (0 -> 199 999 999)
Data collection 1*
Parallel view [2 (default/default)]
Time
Domain
Recv bytes
Sent bytes
Method
Dst IP
694 130 (0 %) / 200 000 000
Zoomed view [2 (default/default)] on axis Domain]
(No correlation)
Layer stack [2 (defaul...
freq IP
New layer 0
Zoomed view [2 (default/default)] on axis Dst IP]

Listing [2 (default/default)]

	Time	Domain	Recv bytes	Sent bytes	Method	Dst IP	Protocol	Subdomain
165563881	10/Jun/2010:1...	216.115.209.254	273	125	CONNECT	192.27.201.11		
165563946	10/Jun/2010:1...	216.115.212.254	273	125	CONNECT	192.27.201.11		
165565118	10/Jun/2010:1...	216.115.209.254	273	125	CONNECT	192.27.201.11		
165565176	10/Jun/2010:1...	216.115.212.254	273	125	CONNECT	192.27.201.11		
165565974	10/Jun/2010:1...	216.115.209.254	273	125	CONNECT	192.27.201.11		
165566051	10/Jun/2010:1...	216.115.212.254	273	125	CONNECT	192.27.201.11		
165572222	10/Jun/2010:1...	216.115.217.254	273	125	CONNECT	192.27.201.11		
165572465	10/Jun/2010:1...	216.115.216.250	273	125	CONNECT	192.27.201.11		
165573510	10/Jun/2010:1...	216.115.217.254	273	125	CONNECT	192.27.201.11		
165573572	10/Jun/2010:1...	216.115.216.250	273	125	CONNECT	192.27.201.11		
165574606	10/Jun/2010:1...	216.115.217.254	273	125	CONNECT	192.27.201.11		

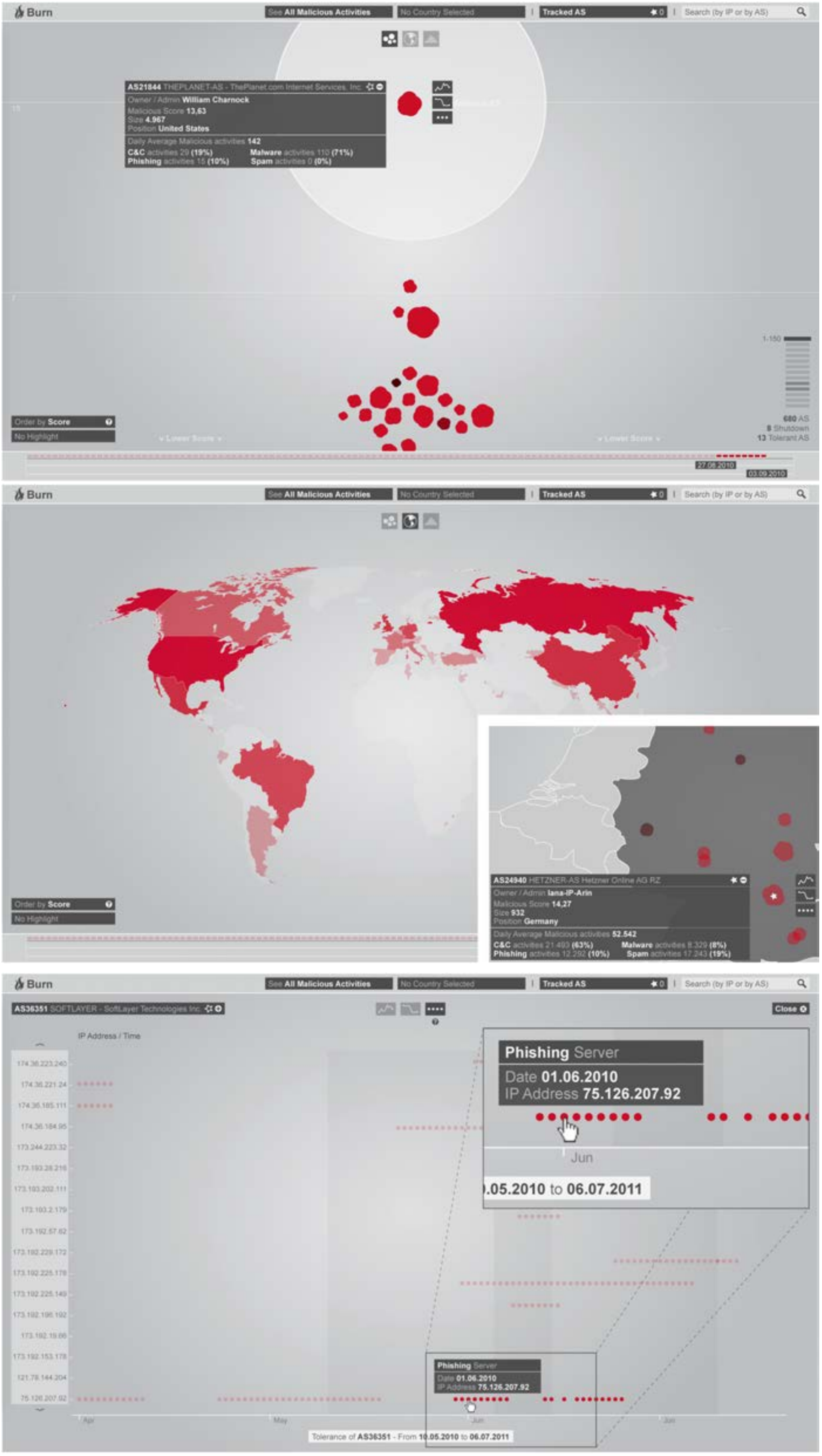
Figure 2.23: PicViz [71] is designed around a central parallel coordinate plot which can be configured to show correlations in data between dimensions.

and passing analysis notes from one tool to the next. However, even if some tools take multiple sources of data into account, they concentrate on a single visual representation regardless of the nature of the data, and pursuing the analysis by traversing from one dataset to another is not addressed.



text: the red stuff. Half-way up the hill he met Bagheera with the
names: Bagheera

Figure 2.24: SeeSoft [16] displays log files in a "10,000 feet view" and applies a custom colour code to each line to help the user detect patterns and groups of events.



AS21844 THEPLANET-AS - ThePlanet.com Internet Services, Inc.
Owner / Admin William Charnock
Malicious Score 13,63
Size 4.967
Position United States
Daily Average Malicious activities 142
C&C activities 29 (19%) Malware activities 110 (71%)
Phishing activities 15 (10%) Spam activities 0 (0%)

AS24940 HETZNER-AS Hetzner Online AG RZ
Owner / Admin Iana-IP-Arin
Malicious Score 14,27
Size 932
Position Germany
Daily Average Malicious activities 52.542
C&C activities 21.493 (63%) Malware activities 8.329 (8%)
Phishing activities 12.292 (10%) Spam activities 17.243 (19%)

AS36351 SOFTLAYER - SoftLayer Technologies Inc.

Phishing Server
Date 01.06.2010
IP Address 75.126.207.92

.05.2010 to 06.07.2011

Tolerance of AS36351 - From 10.05.2010 to 06.07.2011

Figure 2.25: In BURN [60], systems are represented by bubbles, more animated and irregular when active and which rise to the top of a bubble chart the more they are active. Sparkline graphs enable quick inspection of activity and behaviour changes, while a histogram shows global activity over time and enables time frame filtering. Two other views, a global map and an activity matrix, enable both a geographical inspection of activity and more accurate inspection of changes in activity.

2.3 VISUALISATION FOR REPORTING

For a team to function properly, its members need to be coordinated, especially by sharing information. Depending on how the team is organised and the roles of each member this sharing of information can take multiple forms.

When team members assume the same roles and work together, they need to maintain a shared state for the current tasks. This keeps each member up to date so together they can accomplish it completely and efficiently. Their synchronisation requires a basic minimum: sharing state updates with references to their context. For team members with different roles, working together requires sharing more context to understand received messages. For example, when anomalies are flagged during monitoring and need to be shared with analysts for further study, the contents of the task needs to be accompanied by context within the data as well as an explanation.

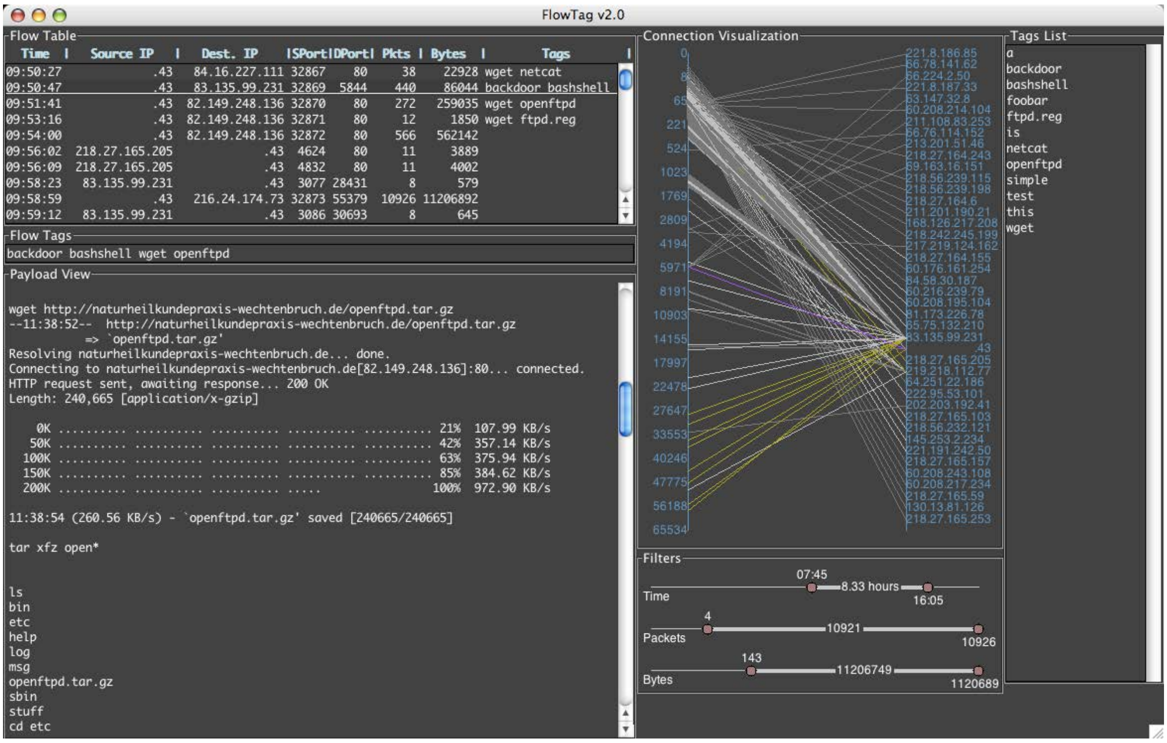
Some pieces of information also need to be sent outside the team to people who might have little pre-existing context for understanding. These people can range from stakeholders with some knowledge of the situation to the public who can quite possibly have no knowledge of the domain. They require much richer context, and may even need a full explanation to understand the contents of the message. The key in these situations is to send the right amount of context, essential for understanding information presented within reports. When a destination is closer to the problem, the amount of context needed for the message to be understood is much less than if the destination has little or no prior context for understanding.

To elaborate, we now explore both ends of this spectrum, designated as *collaboration* and *communication*.

2.3.1 Collaboration

The aim of collaboration is to divide a task into parts among people and synchronise them so they progress faster. Monitoring operators often need to collaborate as teams. Whether these operators are working in parallel or relaying in shifts, sharing metadata is an essential part of the job. For example, operators will want to mark events as benign or dangerous and avoid unnecessary work. During analysis phases, tagging and linking information is essential. Not only does this metadata provide the contents of the message, it can also be linked to the context in which it was tagged. Reports with the right context are necessary for effective mitigation or for communicating back to operators monitoring the real-time feeds of information.

To manage the results produced by long and complicated network data analysis sessions and help to avoid bad reports, FlowTag [42] lets analysts tag interesting and related items. This makes the process



```
 Time  |  Source IP    |   Dest. IP     |SPort|DPort| Pkts | Bytes  |      Tags
09:50:27             .43   84.16.227.111 32867    80     38    22928 wget netcat
09:50:47             .43   83.135.99.231 32869  5844    440    86044 backdoor bashshell
09:51:41             .43  82.149.248.136 32870    80    272   259035 wget openftpd
09:53:16             .43  82.149.248.136 32871    80     12     1850 wget ftpd.reg
09:54:00             .43  82.149.248.136 32872    80    566   562142
09:56:02  218.27.165.205             .43  4624    80     11     3889
09:56:09  218.27.165.205             .43  4832    80     11     4002
09:58:23   83.135.99.231             .43  3077 28431      8      579
09:58:59             .43   216.24.174.73 32873 55379  10926 11206892
09:59:12   83.135.99.231             .43  3086 30693      8      645
```

```
backdoor bashshell wget openftpd
```

```
wget http://naturheilkundepraxis-wechtenbruch.de/openftpd.tar.gz
--11:38:52--  http://naturheilkundepraxis-wechtenbruch.de/openftpd.tar.gz
           => `openftpd.tar.gz'
Resolving naturheilkundepraxis-wechtenbruch.de... done.
Connecting to naturheilkundepraxis-wechtenbruch.de[82.149.248.136]:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 240,665 [application/x-gzip]

    0K .......... .......... .......... .......... .......... 21%  107.99 KB/s
   50K .......... .......... .......... .......... .......... 42%  357.14 KB/s
  100K .......... .......... .......... .......... .......... 63%  375.94 KB/s
  150K .......... .......... .......... .......... .......... 85%  384.62 KB/s
  200K .......... .......... .......... .....                100%  972.90 KB/s

11:38:54 (260.56 KB/s) - `openftpd.tar.gz' saved [240665/240665]

tar xfz open*


ls
bin
etc
help
log
msg
openftpd.tar.gz
sbin
stuff
cd etc
```

Figure 2.26: FlowTag [42] provides classic command line tools along with sliders for simple filtering and parallel plot correlation. Additionally flows can be tagged for future reference and for building better reports.

easier to manage, but also easier to share with others, making the analytical process a more collaborative task. Tagged data also makes full reports easier to produce, thanks to the possibility of gathering all related data into one corpus (Fig 2.26).

Additions to NVisionIP [41] also add the capacity to record visual exploration paths. This means reports can take the form of a replay of the actual exploration.

2.3.2 Communication

When reports are transmitted to people outside of a team, the recipients often require more context than internal people. These people can be close, such as other teams within the same institution, but can also be as distant as remote stakeholders or journalists.

For these closer people who already have some situational context, sharing the visual state can suffice. For example, VisAlert [21], [46], [47] is designed to provide essential situational context. When an alert needs to be reported a screen capture can suffice because it contains all the information required to understand why that alert was reported (Fig. 2.9).

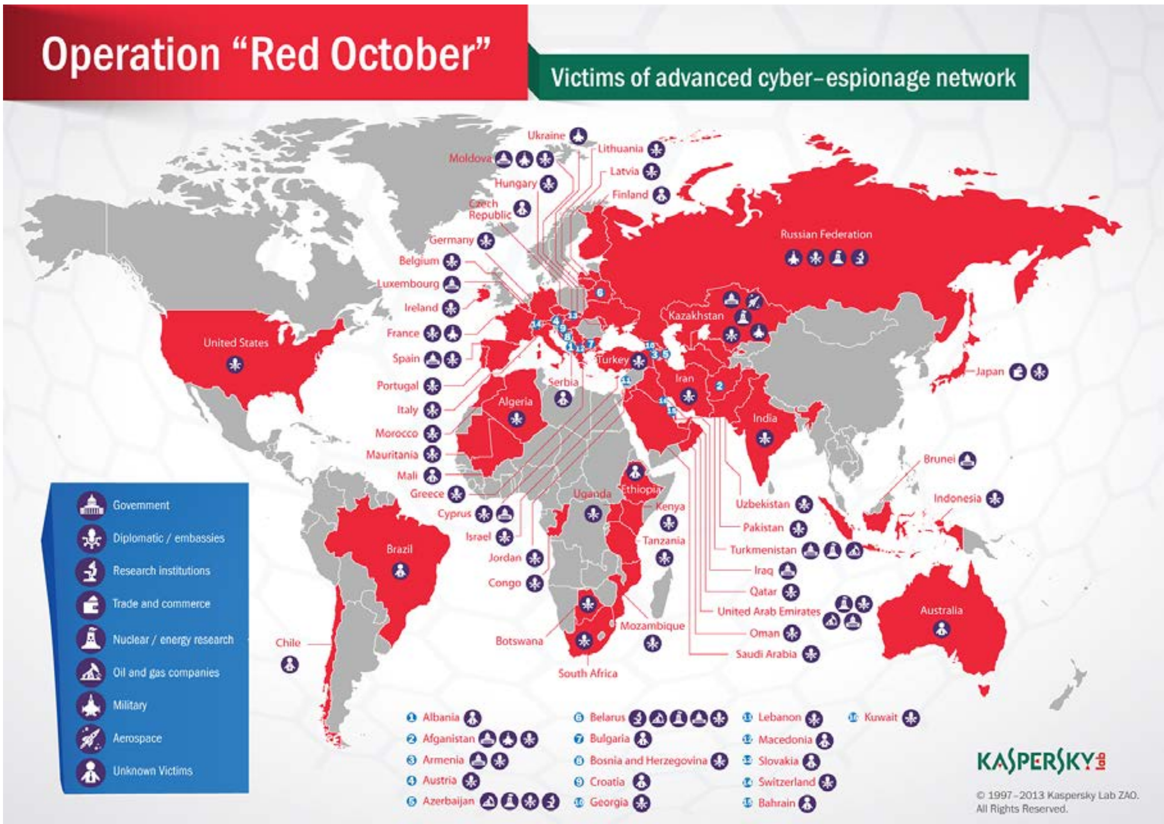


Figure 2.27: This press release from Kaspersky Lab maps out the world wide victims of the Red October malware.

In cases where the recipient is new, the message might need to be accompanied by a richer, more complete context. In these situations, the message can make use of pre-existing shared context. Visual metaphors are an invaluable source of shared context, information which all parties are familiar with. Maps are often used for representing simple datasets, but deliver a whole host of familiar context to better understand those few points.

Other visualisations, while not the most efficient for correlation or analysis, are less abstract and therefore better at communicating meaning and the nature of the information. Node-link graphs for example, while impractical for many cases, are effective at communicating the idea of a interconnection.

To summarise, tools which help operators with reporting allow them to share information between each other when collaborating in a team, but also communicate information outside of the team to third parties. By providing features which help operators to collect and annotate important data, the necessary information for collaboration is more readily available. When directly linked to the relevant context, this information is also more easily shared with people who may not have the knowledge necessary to directly understand the issues at hand.

2.4 CONCLUSION

In this chapter, we have presented and analysed many visualisation tools dedicated to security event visualisation to associate techniques with the problems and goals faced by operators in three different situations. *Monitoring tools* often use a fixed visualisation to focus on changes in data, looking for patterns. *Analysis tools* help to search data using more configurable visualisations for better exploration of data. *Reporting tools* extend monitoring and analysis tools with features for taking notes and sharing relevant information within teams or transmitting it to external parties.

Together these tools cover a wide range of situations and help to address many different problems. It would seem that if all of these tools were readily available security issues would be easily dealt with. However, incidents are often combinations of issues from multiple different systems each providing different sources of information to process. Effectively handling such a breach would require choosing the right tools, but with so many options the risks of choice aversion [61] are high. Added to this are the difficulties of moving results from one tool to another when points of interest require a different point of view.

Because these tools tend to share visualisation approaches for similar data, we advocate that the choice should be between representations instead of between tools. The following chapter describes ELVis, a tool we designed to assist with security visualisation and help practitioners concentrate on their information instead of choosing representations.

3 ASSISTIVE SECURITY VISUALISATION

While they are invaluable sources of information, log files are often impractical to analyse both because of their human-oriented formats and because of their size, many gigabytes being not uncommon.

Many visualisation tools, among which [16], [39], [53], [71], have been designed to help analysts better understand the content of log files. These tools often focus on a specific type of log file or a specific set of log files. Due to that fact, they do not allow the analyst to opportunistically benefit from other types of log files that could be available.

Building visualisation tools requires visualisation expertise and knowledge of domains such as statistics, design and psychology. Unfortunately, security experts trying to create ad hoc visualisation tools are rarely able to complete the task well due to lack of experience. They are first and foremost security experts, and second – if at all – experts in visualisation. History has shown that although visualisation without training can sometimes be innovative and effective, it can also lead to misleading results [75].

Using security visualisation software requires knowledge and experience in security, especially for analysing custom log formats and system configurations [53]. When faced with a situation, experts rely on familiar processes to ascertain the situation, identify issues, and find solutions. With these sequential goals in mind, they intuitively develop habits in the form of behaviour patterns and personal protocols [2].

In our case, the tasks at hand require exploring multiple different sources of information, but also unfortunately choosing a specific tool every time a new source of information needs to be explored.

In light of these observations, it seems reasonable to advocate that security visualisation should let security experts concentrate on their objectives as much as possible while freeing them from issues outside their area of expertise.

To this end we designed ELVis, a security-oriented extensible log visualisation tool. ELVis allows security experts to import log files with multiple formats (e.g, apache standard logs and syslog files such as authentication logs) and explore them through relevant representations automatically selected and generated according to the data chosen for display.

ELVis was built with two objectives in mind:

1. to allow security experts to benefit from appropriate visual representations of the log files they need to analyse without the need for experience in additional domains such as graphic design.
2. to be as versatile as possible by handling many types of log files and by being extensible for handling extra formats without requiring the help of visualisation experts to design new representations specific to these new types of log files.

The interaction process with ELVis is composed of three phases:

LOG ACQUISITION AND SUMMARY VIEW, during which the user imports the required log files. The resulting datasets are then automatically enriched by the system and summary visualisations are displayed for each of them using small multiples [74], [75]. These representations are the starting point for exploring the logs.

DIMENSION SELECTION AND REPRESENTATION GENERATION, during which the user selects fields to be represented and relevant representations for these are automatically chosen.

FILTERING, during which the user can remove unwanted information or, equivalently, focus on wanted information.

Subsequent iterations and enhancements also added a phase of **brushing and filtering,** during which the user can brush the timeline to focus on periods of time or select specific values of interest in the visualisations to inspect related events.

These three phases are described in detail in the rest of this chapter. In Section3.1, we explain how logs are managed in ELVis. In Section3.2, we illustrate the first representation proposed in ELVis, which summarises a dataset. In Section3.3, we describe the user interactions and show how valid representations are automatically chosen according to the information the user wants to display. In Section4.3 we discuss how the main features of ELVis were implemented. Finally, in Section4.4, we demonstrate the previous functions using the Honeynet Forensic Challenge 10 dataset [69].

3.1 LOG MANAGEMENT

Numerous security-oriented visualisation tools are specific to a certain log file format. For instance, ClockView [37] and PeekKernelFlows [77] use NetFlow logs to monitor large IP spaces over long periods. NetBytes Viewer [67] also uses NetFlow logs to focus on the communications of a single host/subnetwork, and is implemented in the more recent project FloViz [68], which offers more visualisation options. TNV [25] represents pcap files in a way that allows better comprehension of the communications between hosts on a network. Finally, [1] provides a representation to make it easier to understand StealthWatch IDS logs. In contrast with these proposals, ELVis is designed to provide relevant representations for a large set of security-related log files.

The log import process is consistent with the first steps in the pipelines described in Section1.1. In this section, we first describe how the log files we took into consideration are organised. We then describe the acquisition, parsing and mining parts of our data pipeline by showing how ELVis automatically extracts, structures and enriches datasets from these logs, while adding metadata to provide more information for the end users.

3.1.1 Log file organisation

Log files are automatically generated by operating systems or applications to keep track of events. When an event occurs, a new *entry* is added at the end of the log file. Without loss of generality, we consider that each *entry* is stored as a line in the log file. Consequently, each event of interest corresponds to a line. We also consider that all the entries/lines in a log file follow the same structure: each of them contains the same *fields* (or columns). Listing 1 displays a sample log of Apache access. For every *entry* in a log, the value of a *field* may be null (not present) or undefined (with no value). For example, in listing 1 the identd and userid fields are not informed, and are instead filled with the - placeholder. This definition of logs is very broad and encompasses many data sources such as tcpdump packet capture files, Snort alert logs, Apache access and error logs, Linux authentication logs, etc. However, it does not cover certain log files well such as the Linux kernel log for which the format is more permissive, and Dpkg logs, for which each event can span several lines.

```
10.0.1.2 - - [19/Apr/2010:07:33:47 -0700] "GET /feed/
↪  HTTP/1.1" 200 16605 "-" "Apple-PubSub/65.12.1"
10.0.1.2 - - [19/Apr/2010:08:02:10 -0700] "GET /feed/
↪  HTTP/1.1" 200 16605 "-" "Apple-PubSub/65.12.1"
123.4.59.174 - - [19/Apr/2010:08:26:30 -0700] "GET
↪  http://proxyjudge1.proxyfire.net/fastenv HTTP/1.1" 404
↪  1466 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT
↪  5.1)"
```

Listing 1: A sample standard apache log file.

In log files, each *field* can be defined by its *meaning*, a *type* or an *extendable type* and its *value*. In ELVis, the *meaning* is provided by a label which describes what the field means. For instance, it can be `Source IP`, `Destination IP`, `Source port`, `Time`, `Alert Severity`, `HTTP Response Code`, etc. In different log files, the fields can have very different meanings since each log file contains information which is relevant specifically to the application or system which created it. Furthermore, some meanings can be very specific to an application and will not be used in any other type of log file, making it very difficult to be exhaustive when defining possible meanings. In consequence, the meanings of the various fields are mainly of interest to the user and will not be used in ELVis to automatically select suitable representations.

In ELVis, each field is also associated with a *type*. In contrast with the meanings, the types we define are generic enough to encompass all the fields currently in use for the log files we selected, while being specific enough (as we show in Section3.1.3) to allow the automatic selection of relevant representations. We currently use three different basic types:

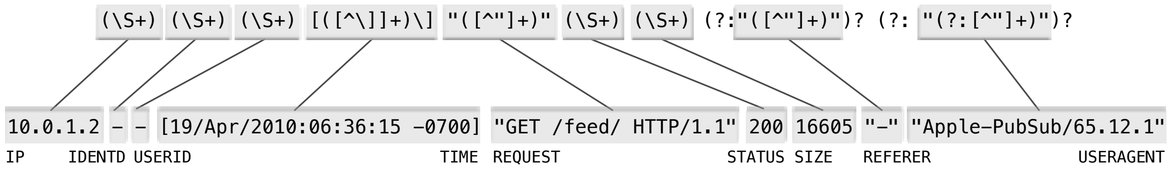
CATEGORICAL when two *entries* with the same value for the field belong to the same category. Values in categorical type fields cannot be naturally ordered or added. However, a set of unique values can be computed (i. e., how many categories there are), as well as their distributions, etc. `IP address`, `UDP/TCP port` or `CVE value` are examples of categorical `fields`.

QUANTITATIVE when the values of the field can be counted, added, compared and can have computable means and sums. For instance, a `packet size` field is quantitative.

TIME when the values of the field are time, which can be presented in multiple different ways: timestamps are numbers, whereas dates are often strings.

GEOGRAPHICAL when the values can be located on a map. For instance, the `location` field that gives the GPS coordinates of an IP is geographical.

Some fields are very common in security-related logs, e.g, `timestamps`, `IP addresses` and `ports`. These kinds of fields have a specific meaning for security experts and are generally used to obtain further information. For example, GPS coordinates can be deduced from a public IP address, the weekday can be deduced from a timestamp, etc. In consequence, an *extendable type* can be associated with these specific fields in addition to one of the three fundamental types. As their name implies, extendable types are used to automatically enrich the dataset by creating new fields which depend on a field having a



```
(\S+) (\S+) (\S+) [([^\]]+)\] "([^"]+)" (\S+) (\S+) (?:"([^"]+)")? (?: "(?:[^"]+)")?

10.0.1.2 - - [19/Apr/2010:06:36:15 -0700] "GET /feed/ HTTP/1.1" 200 16605 "-" "Apple-PubSub/65.12.1"
```

Figure 3.1: Each group in a format pattern is associated with a field in a log line.

given extendable type. Currently, we have defined four extendable types: `timestamp` which can become human readable time, `IP` which can be geolocated, `TCP port` and `UDP port` which can be associated with default services. We provide more information on how extendable types are used in Section3.1.3.

In the following sections, we explain how datasets are acquired from log files, how (extendable) types are assigned to fields and how datasets are augmented. Later in this thesis, we show how field types are used to choose the relevant representations.

3.1.2 Log file acquisition

The first step of log file acquisition consists in parsing the log files submitted by the user. (S)he can submit log files either by dragging them into the browser window or by using `upload` button on the top right of the page (cf. Fig. 3.2 p.62).

ELVis is configured with a set of dataset formats (among which `Apache standard`, `syslog` and its variations - `auth.log` for instance - and `Snort IDS`). Each of them is linked to a given log file format. When a log file is submitted, ELVis tries to match the first line with each loaded format using its pattern. If a match is found, a dataset is created from the file using the matched format. The format identifier is then added to the dataset, this information being known by the parser. We should mention that in order to create a new dataset format and import new log files, the user need only provide a regular expression matching the format of a line for this log as well as the label, the type and/or extendable type for each field (Listing 2).

This approach works well for many log formats, but has limitations. Optional fields can be managed by using optional groups in the line pattern, but malformed files are a challenge. Regular expressions depend on strict matching, and although logs are generated automatically in a consistent manner, file corruption during storage or transmission can remove separators or tokens essential to matching fields. Any line which does not match is skipped, which provides some measure of fault tolerance. However if this line happens to be the first line, used for selecting a compatible format, no matching format will be found. If it contained security relevant information, the whole log entry is lost and will not be taken into account. Multiline logs are also problematic, and would require an approach using state machines instead of stateless regular expressions to match tokens for the start and end of events.

Once the whole file has been parsed and associated with a labeled and typed dataset format, the augmentation stage can start.

3.1.3 Log file augmentation

ELVis augments logs in two different ways: *"horizontal" augmentation* and *"vertical" augmentation*.

Horizontal augmentation consists in adding extra fields to each *entry* (in other words, new columns to the dataset) based on the extendable types. If an `extendable type` is found, the corresponding fields are generated for each *entry* of the dataset. For instance, a `location` field is added for each field exhibiting the `IP` extendable type in the *entry*. This new field is automatically labeled "`Loc[`*labelOfTheIPfield*`]`" to indicate this extension.

When horizontal augmentation is completed, ELVis performs *vertical augmentation* on the dataset. This consists in computing statistics for each field (including the fields that have been added during the horizontal augmentation phase) which are global to all the values. Vertical augmentation of a field depends on the type of the field. On `categorical` fields, a set of unique values is computed as well as their distribution (see table 3.1). On `quantitative` fields, minimum, maximum, mean and sum total values are computed (see table 3.2). The computed results are stored as metadata in the dataset.

value	200	301	302	400	404	500	502
count	306	8	13	3	29	4	2

Table 3.1: For categorical fields such as HTTP status codes, unique values and their distribution are computed.

statistic	sum	min	max	mean	median
value	4682912	20	16605	13773.27	8292.5

Table 3.2: For quantitative fields such as HTTP response sizes, the sum, minimum, maximum, mean and median values are computed.

Each of the imported log files is then displayed in a summary view, which are described and explained in the next section.

```
Elvis.registerFormat({

  name: ’apache log’,

  pattern: /(\S+) (\S+) (\S+) \[([^\]]+)\] "([^"]+)" (\S+)
   ↪ (\S+)(?: "([^"]+)")?(?: "(?:[^"]+)")?/,

  fields: [
    {
      label: ’ip’,
      type: [’categorical’, ’ip’]
    }, {
      label: ’identd’,
      type: [’categorical’]
    }, {
      label: ’userid’,
      type: [’categorical’]
    }, {
      label: ’time’,
      type: [’time’],
      transform: d3.time.format(’%d/%b/%Y:%X %Z’).parse
    }, {
      label: ’request’,
      type: [’categorical’, ’http-request’]
    }, {
      label: ’status’,
      type: [’categorical’, ’http-status’]
    }, {
      label: ’size’,
      type: [’quantitative’],
      unit: ’bytes’
    }, {
      label: ’referer’,
      type: [’categorical’]
    }, {
      label: ’useragent’,
      type: [’categorical’]
    }
  ]

});
```

Listing 2: The format definition for the apache standard log format. Each format contains a name, a regular expression for parsing the intended logs and an ordered list of fields which map to the log file (see Fig 3.1). Each field contains a label and a list of types and any possible extendable types.

3.2 SUMMARY VIEW

When dataset augmentation is finished, the next step in the data pipeline is to build a first representation. For each dataset a visual summary of the log file is displayed. Fig. 3.2 shows a global view of the ELVis interface after two log files have been imported: an apache log file and an auth log file.

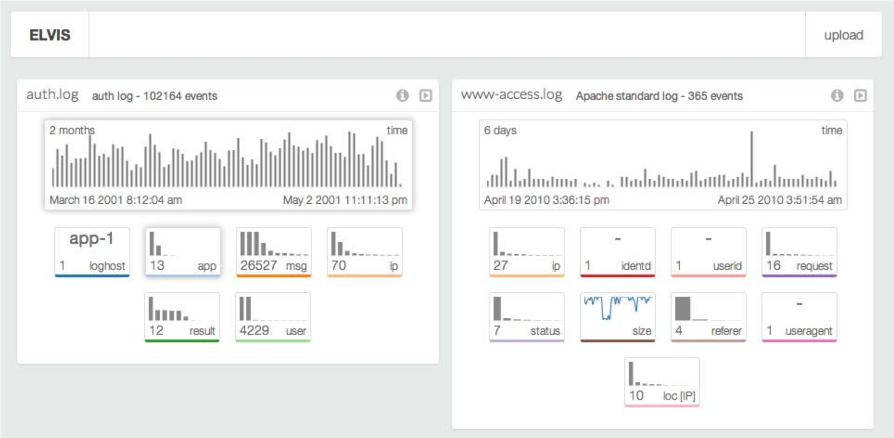


Figure 3.2: The ELVis interface after two log files have been imported.

The summary view provides the user with the global overview of a dataset. Figure 3.3 shows the summary view of an apache log file.

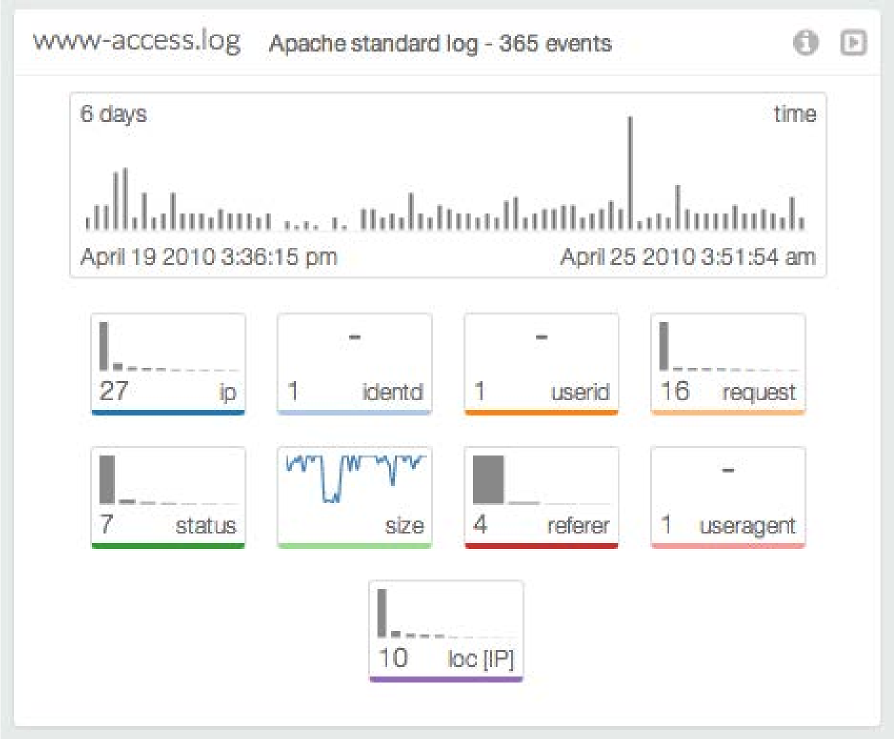


Figure 3.3: The summary view of an apache log file in the first iteration of ELVis.

The name of the file (`www-access.log`), the type of the log file (`Apache standard`) and the number of events (`365`) are displayed in the view header at the top. To its right, an information icon allows the user to obtain more information about the dataset, i. e., a more verbose description of the format, how the information was obtained and the size of the dataset. The arrow icon allows the user to reduce the summary view so as to save space on screen.

The next component is a chart in the upper part of the view which displays event distribution in the dataset over time. The duration, start and end times are displayed in corners and the bar chart displays the evolution of the number of events across time. This provides the user with a global overview of the considered period as well as of the distribution of the events to rapidly pick out unusual points in time during which too many or too few events occurred.

Under the main visualisation, smaller blocks display sparkline summaries for each field of the dataset. The label for the field is displayed in the bottom right of the block and the content of each block depends on the type of the field that this block represents. For `categorical` fields (such as `IP` in Fig. 3.3), the number of categories is displayed in the bottom left and a sparkline type bar chart displays the distribution of the various values[7]. In some specific cases, a given `categorical` type field may have a single unique value for all the *entries* in the dataset. This for instance is the case for our Apache access log where the fields `identid`, `userid` and `useragent` are all set to a default value in the log file, i.e, "`-`". In this case, the single unique value is displayed instead of a sparkline visualisation (provided the value fits within the allotted space). For `quantitative` type fields (such as `size` in Fig. 3.3), a sparkline type line chart displays the evolution of the field across time with the minimum and maximum displayed in a similar way to the `timestamp` type fields for which a user-friendly representation of the beginning and end date are provided.

These field summary blocks provide valuable insights for the user in a concise way. First, the user knows at a glance which fields are available. Second, (s)he can quickly identify which fields might or might not be relevant to the analysis objectives, e. g., the same value for this a given field is present in all the *entries* of the log. Finally, the summary in each small block can help the user know which intervals are considered (the timeframe, for instance) and therefore identify anomalous values, characteristic trends, etc. Although representations (variable extents, distributions and sparklines) are chosen automatically based only on the type of the field, we argue that they are in fact valuable if not concise [78].

In this section, we presented the summary view which is automatically generated for imported log files, provided that the correct parser

[7] This information is pre-calculated when the dataset is first processed and loaded (see vertical augmentation of the dataset in Section3.1.3).

is available. We insist on the fact that when creating a new parser for a new type of log file, the user never has to choose the representations used in the summary, and only needs to specify the types of fields. Consequently, (s)he does not need experience in design or visualisation. In the next section, we show how the user interacts with the summary view to obtain details on the various fields and their characteristics.

3.3 USER INTERACTIONS

In this section, we describe user interactions with the dataset through the summary view. We begin by describing how the user chooses the fields (s)he is interested with. We then explain how representations are automatically selected according to the types of the selected fields. Finally, we briefly describe how representations are generated.

3.3.1 Selecting fields of interest

As shown in Section3.2, each field of the dataset is presented as a summary block in the summary view. To interact with the dataset, the user can select one or more blocks to form a simple subset of the fields (s)he is currently interested in. A shadow then surrounds these fields, providing visual feedbacks on the fields that are selected.

In order to obtain a detailed representation of the selected fields, the user drags them off the summary view and drops them on a free space of the background of the web page. While this mode of interaction does not display any affordance for discovery by first time users who have not been briefed, we found that the metaphor of dragging fields of interest off the summary views to obtain detailed views was acquired very quickly. Furthermore, given how representations are placed on the screen, it is very easy for the user to find a free space without having to go through a complex process.

Once the selected fields are dropped, ELVis automatically selects a relevant representation for the data. The way this selection is performed is described in the next section.

3.3.2 Automated selection of representations

ELVis was designed to assist with the selection of adequate visualisations based on context and objectives.

For individuals having no specific skills in visualisation, the selection of a relevant representation for a dataset is a complex task. Even if Bertin [5] and Wilkinson [78] have provided fundamental pieces of work on this topic, one cannot expect security experts to spend extensive time on selecting a representation for the data they want to inspect. As a consequence, ELVis automatically selects a relevant visualisation given the fields the user wants displayed.

To that end, based on the type of each selected field, ELVis selects a relevant representation according to the following rules, inspired by [5], [78]:

- Selecting a single `categorical` type field which number of unique values is smaller than eight[8], a pie chart is produced.
- Selecting a single `categorical` type field which number of unique values is larger than seven, a bar chart is produced.
- Selecting a single `quantitative` type field produces a detailed histogram displaying the general distribution of its values.
- Selecting a single `geographical` type field produces a map with plotted points and an adjusted view to encompass them.
- Selecting two `categorical` type values produces a matrix based adjacency chart for correlation.
- Selecting two `quantitative` type fields produces a scatter plot for finding correlations. However, if one of these fields is of `time` type, this produces a line chart to display value evolution.
- When any other combination is chosen, ELVis falls back to parallel coordinates *à la* PicViz [71], that is known to be able to display numerous values and multiple fields.

Concretely, these rules are functions which filter out incompatible options from the set of available visualisations. Visualisations are selected based on the selected dimensions and the properties of their data, and only those which pass at least one of the previous rule are kept.

The set of representations available in ELVis has deliberately been kept small. We did not want to take the risk of confusing the user with many possible representations for a single concept, nor create very specific and possibly hard to understand representations that would only appear in very specific cases. One of the objectives of ELVis is to be extensible to new kinds of log files formats with fields which may be semantically very different from the ones that are already available in the tool. It is therefore important that some generic-enough representations are available to handle these very specific fields.

The chosen representation is then scaled and configured based on the values computed during the vertical augmentation for each field. Finally, the representation is created and displayed on the screen. Fig.

[8] We experimentally found that when there are more than seven values in the category, the pie chart becomes unclear.

3.4 shows an example of various representations obtained from the apache log. The pie chart on the upper right shows the proportions of the various values in the `categorical` type field `status`. The adjacency matrix on the bottom left displays the relations between the two `categorical` type fields `statuses` and `IP`, showing which `IP` caused which `statuses`. Finally, the map on the bottom right corner shows the locations of the values of the `geographical` type field loc[IP].

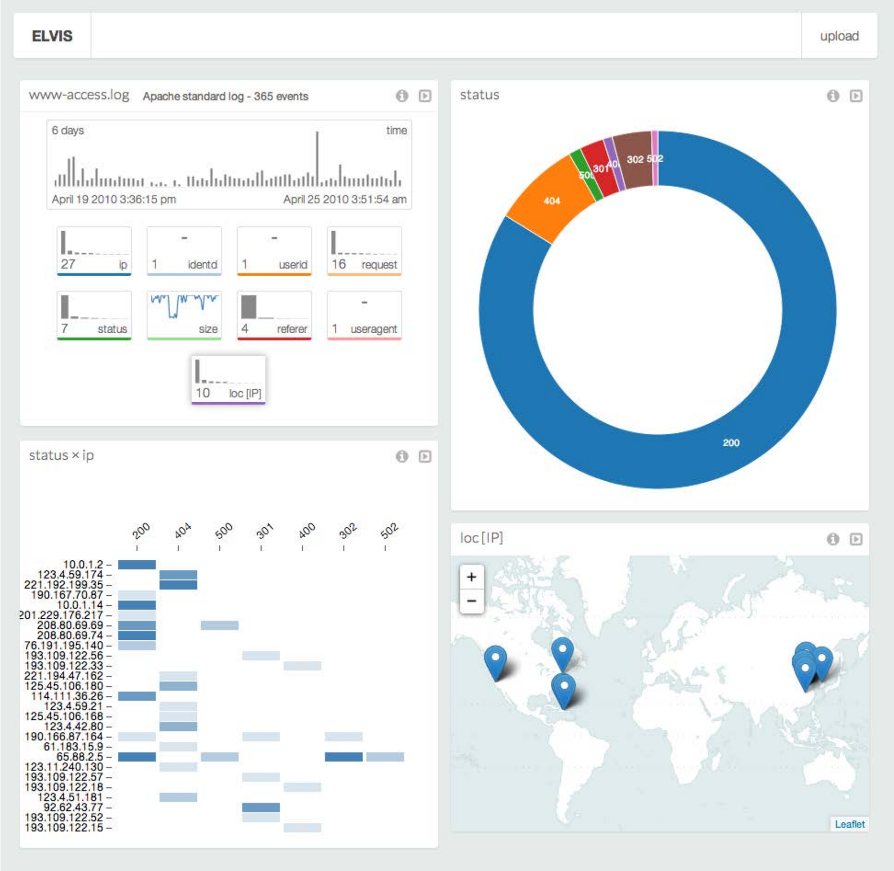


Figure 3.4: Multiple representations automatically selected by ELVis based on the fields chosen by the user. The `status` field is `categorical`, has few values and is represented using a donut chart. When selected with `IP`, another `categorical` type field, a matrix chart is produced. The `IP` field has been extended to add a `geographical` type `loc[IP]` field, which when selected produces a map.

3.3.3 Brushing and filtering

Further work on ELVis led to enhancements to the analysis such as the capacity to interact with the visualisations through brushing and selection. Corresponding to the last step in the data pipeline which concerns interaction, these actions are also responsible for the feedback arcs in the visualisation pipeline. They trigger filtering for the related dimensions and updates to the displayed information (Fig. 3.5).

Brushing the timeline for a datasource filters out events outside the selected time period and focus on those events. Dragging the selection updates the filter and enables time "scrubbing" and enables replaying or searching for large scale events.

Selecting single marks in the visualisations filters out events which are not concerned by the selection and helps people focus on those combinations of dimensions and values. For instance, in the dimension summaries each bar is mapped to a value of the corresponding dimension and its total count. Selecting any bar in a `status` summary focuses on the associated `status` and filters out all data unrelated with that `status`.

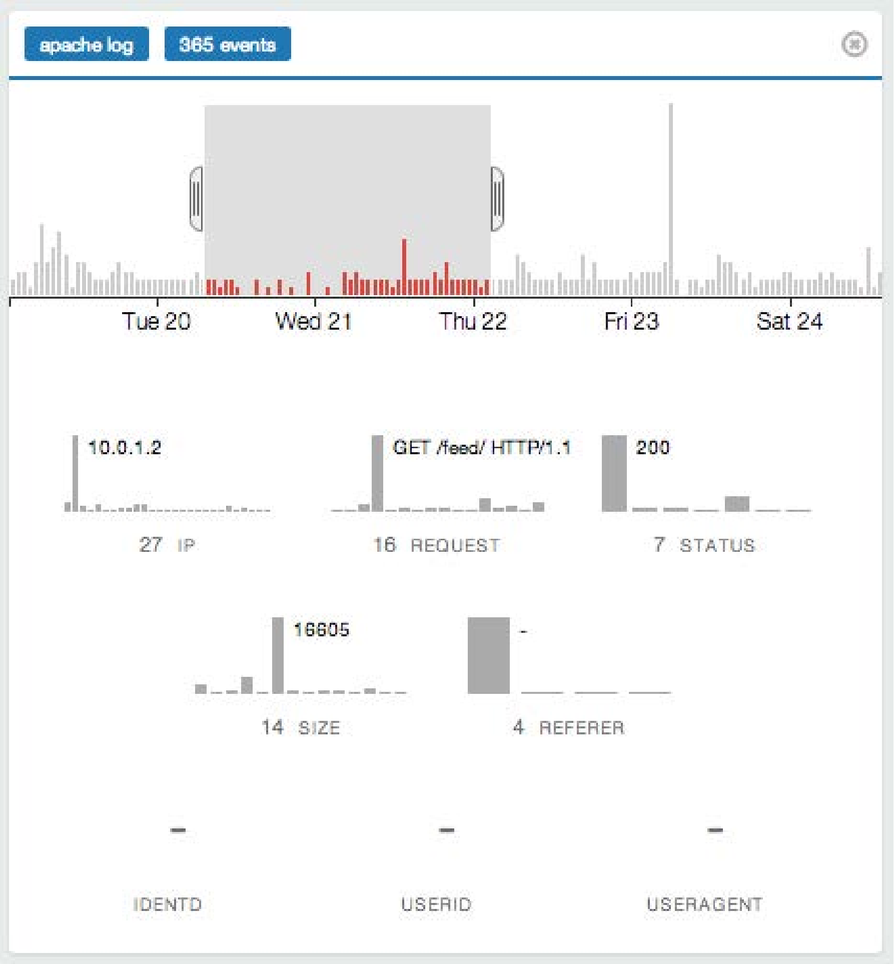


Figure 3.5: Exploring a dataset through brushing and selection. Datasets are distinguished by colour and filtering is available for every visualisation through selection or brushing.

3.4 IMPLEMENTATION

ELVis is implemented for the browser platform with web technologies such as HTML5, Javascript, CSS and SVG[9]. The representations are built using the D3 Javascript library [10] which enables flexible yet semantically meaningful creation of data visualisations.

The first publicly presented version of ELVis used the third party D3.chart library from the Miso project [70] which helps define structures to better build reusable and extensible charts, and Miso Dataset which aids with parsing and structuring datasets.

Following tests, performance was marked as an important issue and ELVis clearly needed optimisation. While there were no problems with handling log files no longer that a few thousand *entries*, ELVis took up to a few minutes to handle files containing hundreds of thousands of *entries*. In a second iteration, Miso Dataset was replaced by Crossfilter [13] to help speed up the import phase and provide a means for filtering the stored datasets. Because Crossfilter is an optimised OLAP cube for aggregation and filtering data, using it for storing and accessing the log files also contributed to a responsive interface with datasets based on slicing and dicing dimensions.

The first version of ELVis was also lacking a mechanism allowing people to visually filter *entries* according to specific values in specific fields. Further work towards implementing selections in the various detailed views added brushing and zooming capabilities to help users explore and drill-down into data to truly make use of each representation. This next iteration of ELVis (Fig 3.5) generated representations which were synchronised with selections in the other visualisations. This aspect was a strong argument towards limiting the number of representations in ELVis: each new representation needed a relevant selection mechanism, which could be difficult to manage in very complex representations.

3.5 EXPERIMENTATION

In order to test ELVis, we used the 2010 Honeynet visualisation challenge dataset [69]. This dataset contains log files that were gathered on a real compromised machine.

[9] HTML5 is the standard for programming web applications, using the Javascript programming language and Cascading style sheets (CSS) for describing application appearance. Scalable vector graphics (SVG) is the web standard for producing scalable graphics.

3.5.1 Exploring logs

ELVis tries to impose as few decisions as possible on the operator, who can instead rely on experience to freely search for patterns significant to the data and systems.

While exploring the log files provided by the honeyviz challenge using ELVis, we found many interesting patterns, including the one seen in Fig. 3.6 in which user authentications (successful or not) are displayed according to time. In this representation, numerous bursts of authentication attempts made for numerous user names are apparent, with no further attempts for a given user name once the burst is over. These cascade like patterns of similar bursts indicate that the attempts are very probably coordinated brute force attacks.

By using the combined aggregations and filters, groups of events could be isolated and inspected without losing track of the exploration process. Thanks to the the feedback provided by responsive filtering we were able explore datasets much more naturally [45].

3.5.2 Interpretation

During these tests, we observed that most representations that were automatically generated by ELVis made overall sense. At no point was there any need for resorting to complex or custom visualisation: once a specific pattern in a given log file was discovered, users were more inclined to repeat the process again with simple visualisations rather than add more fields to the current one. The tests demonstrated that ELVis is useful to visually explore security log files, allowing people to quickly notice relevant facts.

However, these results also revealed the limitations of this solution.

First, representations were created only by interacting with the dataset summaries. While ELVis was not able to perform this task, a further tool was envisaged which would allow that. Further representations could also be produced from these initial representations, making data exploration a progressive drill-down process.

Also, in ELVis, each dataset (i. e., each log file) was isolated from the others and multiple datasets cannot be combined for exploration even if more that one have been imported. Enhancements to selections could enable them to span multiple datasets to correlate between them and combine information. The same logic applies to the visualisations, which could also foreseeably be combined into composite visualisations.

These two issues required an entire new approach, which ultimately led to the development of an new tool based on the same principles but designed this time around the analytical methods adopted by security analysts when approaching new data. This tool was designed to help pursue an analysis from one dataset to another by using

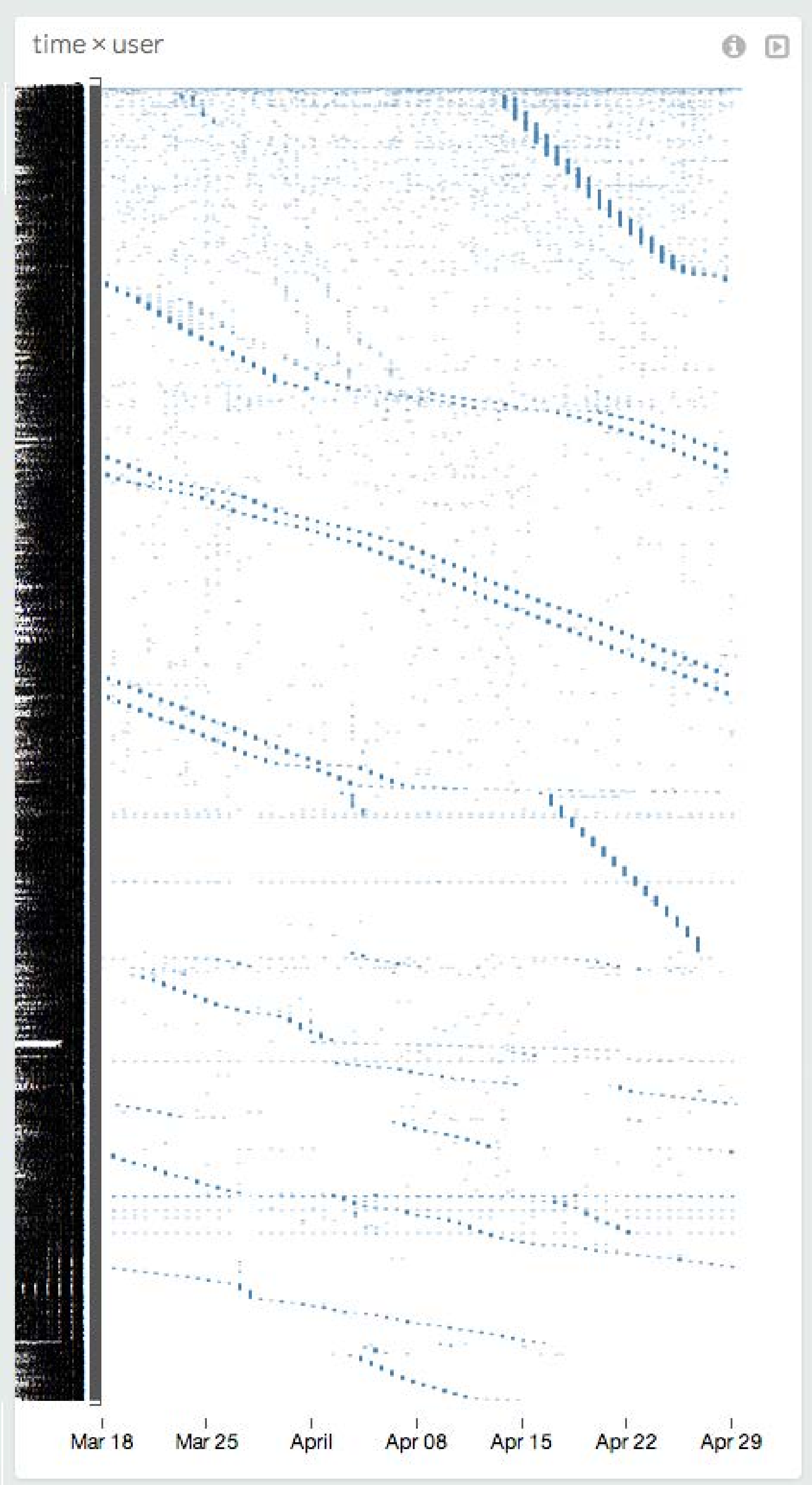


Figure 3.6: The representation of the fields `user` and `time` from the `auth.log` file show distinctive access patterns.

points of interest shared by both and effectively combining multiple datasets within one tool, which we present in the next section.

3.6 CONCLUSION

This chapter discussed ELVis, the proof of concept for a security-oriented log visualisation tool that allows security experts to visually explore numerous types of log files. In order to automatically select relevant representations, ELVis uses information on the type of each field. This way, it can produce concise and expressive summary views for each log file as well as appropriate detailed views for fields selected by the user. If new types of log files need to be analysed, the user needs only provide a regular expression for parsing each line of the log file as well as the type and label for each resulting field.

With this approach, ELVis accomplished its stated objectives:

1. to help security experts to benefit from appropriate visual representations of log files without the need for experience in additional domains such as graphic design.

2. to be as versatile as possible by handling many types of log files and by being extensible for handling extra formats without requiring the help of visualisation experts to design new representations specific to these new types of log files.

Therefore, ELVis allows security experts with no experience of visualisation to benefit from adequate visual representations for their log files. Security experts can input their knowledge of log formats and types into the system without having to specify their visual usage or the outcomes while obtaining valid representations for their log files.

In the next chapter we describe the successor to ELVis. Dubbed CORGI, this tool inherits the same parsing and visualisation mechanisms as ELVis, but with an interface designed for visualising and comparing multiple datasets. CORGI allows the collection of values of interest in datasets for future use in reports but also enables the pursuit of these across datasets, enabling a truly semantic analytical exploration process.

4 INVESTIGATIVE SECURITY VISUALISATION

"Time moves in one direction, memory in another."
— William Gibson

As we discussed in the previous chapter, log files exist in different formats and contain different information depending on the software that generated them and what the file describes. ELVis aimed to explain global trends or detect events that are symptomatic of attacks. In accordance with the Shneiderman mantra, It is often considered that the analysis is a drill down process and that the analyst is searching for a specific piece of information. However, even if detecting each malicious action is fundamental, it is also very important to understand the relations between security events so as to reconstruct the global scenario [43]: once the analyst has found an interesting event, (s)he must be able to discover any other related events, even if these events are found in different log files generated by different sources therefore exhibiting different formats. What are, for example, the causal relations between the attacks in different parts of the system? Having compromised a web server, did the attackers then perform other malicious actions on the system? If so, what are the consequences? In reaction to this, we advocate that IT forensics is an iterative process and that an analyst must be able to easily use information stored in log files to search for related events in other log files, even if these are not *explicitly* related /empha priori.

This chapter presents CORGI, an evolution of ELVis which implements an iterative process inspired by those used for IT forensics by allowing the analyst to use the *values of interest* (s)he finds in a given dataset to filter events in other datasets, thus implicitly relating these datasets and helping to traverse multiple datasets meaningfully.

In this chapter, we first explain how CORGI manages log files. Then, we detail how log files can be efficiently related to each other based on *values of interest*. In Section4.2, we describe the representations and user interactions in CORGI. In Section4.3, we provide some information about implementation. Finally, in Section4.4 we discuss case studies and possible improvements.

4.1 LOG FILES

Inheriting from ELVis, CORGI also uses log files as data sources. In this section, we first present how log files are organised. We then present the various operations that can be performed on a dataset obtained from a log file during an analysis. Finally, we present how connections can be built between *a priori* unrelated datasets.

4.1.1 Log File Organisation

As stated earlier, while log files exhibit various contents and formats, they are generally organised in a similar way. A log file is made of a set of *entries*, each of which corresponds to an *event*. These events are made of a set of *fields*. Since each entry in a given log file corresponds to an event, we also assume that all the log files used by the analyst contain a `timestamp` field which places the event in time. In consequence, most log files can be seen as a table with as many lines as events and as many columns as fields, at least one of them being a `timestamp`.

Re-using the same approach to typing described in the previous chapter for ELVis, each field is designated as either `categorical`, `quantitative`, `time` or `geographical`.

Each field also has meaning and can be associated with a *semantic type*. The *semantic types* in CORGI are different from the previous types in ELVis: they provide information about the nature and meaning of the field. Currently we have identified many semantic types among which: `IP Address`, `TCP Port`, `UDP Port`, `Timestamp`, `URL`, `CVE_ID`, `Size`, `HTTP Method`, `HTTP Status Code`, `FTP command`, `userid`, `City`, `Country`. The analyst can create further types according to the current need. Types should be generic enough to be used in as many log files as possible while being specific enough to avoid ambiguity. For instance, a `Status Code` type is too generic since it could encompass `HTTP Status Code` and `NNTP Status Code`, which in fact do not have the same meaning.

CORGI uses these *semantic types* to create relations between datasets through *categorical* type fields. This aspect will be described in Section4.1.3.

4.1.2 Operations on a Single Log File

When exploring a log file for forensics purposes, an analyst is searching for *events of interest*, events that are of particular relevance for understanding and reconstructing what happened on the monitored system. Different analysts have different approaches for exploring the same data [2]. For instance, they do not start by exploring the same fields. These differences could be explained by the attacks the analysts have been exposed to recently as well as the contextual information each of them has been provided with.

While their ways of exploring a given log file are different, all analysts generally perform sequences of two basic operations: they choose specific fields to be considered and select events according to the values exhibited by these fields. In accordance with the terminology used in relational algebra [11], we call these operations *projections* and *selections*. *Projections* consist in choosing *fields of interest* (for instance, `HTTP Status Code`) so as to focus on the information they contain. *Selections* consist in choosing specific values for a given field. For instance, the analyst chooses all the events in an `apache-access` log file for which the `HTTP Status Code` is equal to `500`, `501`, `502`, `503`, `504` or `505` which correspond to types of server errors.

The exploration of a log file can be described as sequences of projections and selections which lead to *events of interest*, events which are considered particularly interesting for the analyst. We should mention that the differences between the different *modi operandi* of the analysts are linked to the order in which these operations are performed. However, if analysts reach the same conclusions, they will have obtained the same set of *events of interest* whatever order the projections/selections were in.

When an analyst obtains a set of *events of interest*, (s)he can consider that the values of some of the fields are particularly relevant. We call them *values of interest*. For instance, it could be the `IP Addresses` of the hosts which caused the events of interest to be logged, the `CVE_IDes` of the vulnerabilities which were effectively exploited, etc. *Values of interest* are obtained by projecting the *events of interest* for a specific *field of interest* and therefore constitute a set of values the *events of interest* take for this *field of interest* associated with the semantic type of the field.

In the next section, we show how *values of interest* are used to relate log files.

4.1.3 Relating Datasets

Log files offer local views of events on specific parts of the monitored system. As such, they allow the detection of malicious actions that were performed in its own specific context. However, the purpose of forensics is not only to detect unrelated malicious events but also to reconstruct the global scenario of the attacks which happened on the system as a whole. To that end, it is important to be able to relate the various log files. Since our objective is to allow the analyst to opportunistically use any log file available on the system, we also designed CORGI to offer him or her the ability to dynamically relate log files having no explicit *a priori* relation between them. This is done using both the values of interest and the `Timestamp` fields.

4.1.3.1 *Relations Based on Values of Interest*

Each log entry is made of fields which each have a specific type. Two log files which exhibit fields of the same type can be related semantically since they reference objects of the same nature. For instance, relations can be implied between log files that both have `IP Address`, `TCP Port`, `HTTP Status Code` or `CVE_ID` fields.

Following the idea that forensics is an iterative process (one discovery leading to another), *CORGI relies on values of interest to relate log files*. As stated in the previous section, an analyst who discovers events of interest can extract values of interest from chosen fields. Because these values are typed, it is possible to use them for filtering log files with at least one field of the same type, i. e., to select this field and only keep entries with values listed in the provided values of interest. As a result, the new selection only contains events with values previously defined as being of interest in the first log file.

We should mention that if a log file has more than one field of a given type, the analyst can apply the filter to any number of them. For instance, `IP Addresses` of interest obtained from an apache log can be used to select the source `IP Address` field of a snort log file, its destination `IP Address` or both. Relations can also be performed on the same log file. For instance, if some `IP Addresses` are identified as victims (`Destination IP Address`) in a Snort log, it can be interesting to use these as filters on the `Source IP Address` of the same Snort log to discover compromised machines being used as stepping stones to launch attacks.

As explained, log files are related *a posteriori* in CORGI during the analysis: the security operator decides which fields have interesting values and when to follow these to other log files. Only elements considered interesting in one log file can be related to other log files by using them as filters. Using this approach, we avoid combinatorial explosions which can occur when using *natural joins*[10] between log files [11]. While enforcing type constraints prevents the analyst from making mistakes by relating semantically different data, the proposed mechanism also allows more freedom regarding the possible relations that can be explored. We strongly believe that the analyst should have the last word for when it makes sense to relate two log files since (s)he is better informed on analysis context.

4.1.3.2 *Relations Based on Time*

Time is critical information when reconstructing attack scenarios. In point of fact, events belonging to the same attack scenario are linked in one way or another by their time of occurrence. The know-

[10] We use the term *natural join* in the relational algebra sense, i. e., each entry in a log file is associated with every entry in the other if they have the same value for the selected fields.

ledge that two events are simultaneous or that one happened after the other is particularly relevant for an analyst.

In log files, time is stored by the `Timestamp` field. In contrast with the data types we presented in the previous section, the `Timestamp` type is not categorical. Due to clock drifting in the machines and the time it takes to generate a log entry, two events stored in different log files and corresponding to the same observed event may not exhibit the same `Timestamp`, and two unrelated events may share the same `Timestamp`. Modifying `Timestamp` precision to compensate for this fact is dangerous: First, it is very difficult to arbitrarily estimate clock skews and imprecisions. Second, events which are part of an attack scenario may be spread over long periods. Such is the case with *Advanced Persistant Threats* for example.

For these reasons, CORGI uses *visual correlation* to help the analyst relate log files in time. As will be shown in more details in Section4.2, CORGI offers a synthetic representation in which the various log files are presented on a shared time scale to enable a direct visual correlation allowing analysts to perceive simultaneity and possible causality.

In the following sections, we present and explain our choices for implementing these concepts.

4.2 VISUALISATION AND USER INTERACTION

Having explained the conceptual foundation for how CORGI manages datasets, we now describe the interface and the user interactions we designed to help explore logs using visualisation tools. We first provide an overview of the interface, then some details about its different areas and how the analyst interacts with them.

4.2.1 Overview

The main interface in CORGI (see Fig. 4.1) is divided into 4 panels, each of which has a specific purpose linked to a step in the exploration process. We describe these panels in a counter-clockwise fashion, following our general interaction procedure.

In the top left corner, the header panel houses the *log import button*. The analyst can either click this button to select log files for importing or drag and drop these files from a file explorer. When new log files are imported, they are parsed and appear in the leftmost *time view panel* (1) which displays the event time distribution for each log file. When log files are selected in the time view panel, their fields appear in the adjacent *fields summary view* (2), which displays field distributions using sparkline type bar charts. When more than one

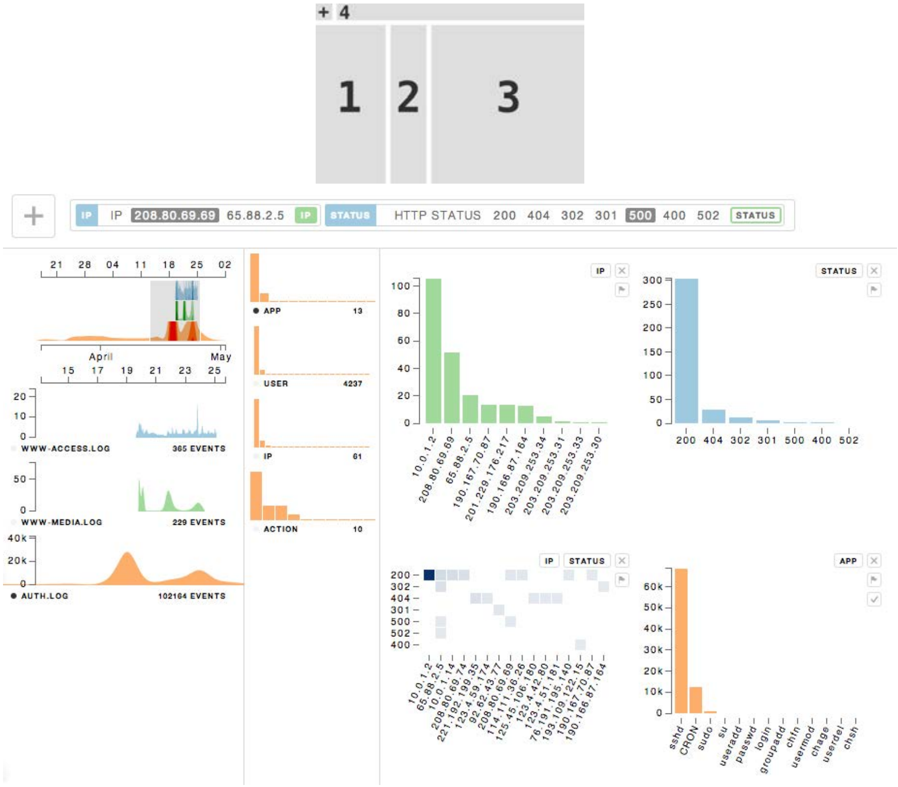


Figure 4.1: An overview of CORGI, with the import button in the top left corner, time view (1), field summary view (2), full-sized chart view (3) and values of interest box (4).

dataset is selected in the time view panel, the fields of every selected dataset are displayed. The *full-sized chart view* (3) is the main panel. It contains all the full-sized charts with axes and labels. Finally, the *values of interest box* (4) located in the header panel is designed both to collect the values of interest discovered during the analysis and to apply these values of interest as filters for other fields of the same type.

An analysis using CORGI follows the same path as our tour of the interface: After the log files have been imported for exploration, the analyst is first given an overview of these to compare the event time distribution. (S)he can then obtain more information about the value distribution for each field and compare these across log files. If some fields look particularly interesting, (s)he may then explore them further and select specific values. When values of interest have been found, the analyst can store them and a new exploration cycle begins: these values of interest can be used to filter fields with the same type or new log files can be imported analysis. This design helps the analyst to avoid losing track of the exploration process and also removes any constraint on the order in which the log files and fields are explored.

We now provide more details about each of these components in the following sections.

4.2.2 Importing Logs

CORGI uses a modified version of the importation mechanism designed for ELVis. The need for a *semantic type* for each field is the only difference with the ELVis log acquisition mechanism.

When importing a set of log files, the first entry of each of them is tested by each available parser until a match is found, in which case each entry is parsed and normalised[11], each field is mapped to its descriptors, the events are counted and the log time period is retrieved.

Each log file is assigned a colour on import which is used in the entire interface for all visualisations related to this dataset to help distinguish between files. This helps analysts identify the source of fields, from which log files values of interest where collected and on which field(s) they have been applied.

For each imported log file, a new representation is created in the *time view panel*.

4.2.3 Timeview Panel

The *time view panel* displays the distribution of events across time for the imported log files. It is composed of two similar representations which both visualise the datasets using stacked charts sharing a scale to enable correlation by time.

The first representations cover a globally encompassing time scale and use a reduced size horizon chart [30]. The dates are displayed in a human-friendly format, which first provides the analyst with the knowledge of the period over which the events have been logged. In Fig. 4.1 for instance, the `auth.log` file spans the full period while the `www-access.log` and `www-media.log` files contain events within a shorted period. In this precise case, this is due to the fact that rollovers are different for `syslog` files and `apache log` files. However, the absence of events over a given period could also mean that the intruder shut the logging system down temporarily, or that parts of the log file have been erased. The horizon-chart based representation allows the analyst to notice these patterns immediately.

This representation also provides an overview of the event distribution over time in a way similar to [20]. As such, macro-events such as DDoS or brute-force attacks for example are detected immediately. Because the representations of the log files are aligned, *visual correlation* is much easier: synchronised attacks over multiple systems, ap-

[11] While `IP addresses` often look the same, `timestamps` for instance exhibit very different formats, e. g., some of them do not contain the year.

pearing in multiple log files, exhibit vertical alignment patterns while causally-related events exhibit delayed activity patterns.

While this representation provides the analyst with an interesting overview of the events, (s)he can also zoom in to obtain more details about a specific period. A *unified brush* on the horizon charts allows for filtering the time period and controls the time scale for the second set of area charts underneath and enables the detailed inspection of our global timeline. Additional information about each log file is also provided: its name, the number of events contained within the selected time period, and a vertical axis for better evaluating the quantity of log entries for each period. The horizontal axis located at the top provides a more detailed timeframe for the selected period.

When the analyst clicks on the representation of a log file, its fields are displayed in the *fields summary view*, which we present in the next section. A second click removes the fields from the *field summary view* to avoid overwhelming him or her.

4.2.4 Fields Summary View

The *field summary view* contains a summary chart representation of all the fields selected in the time view panel. In order to help the analyst, each field exhibits the same colour as the log file it belongs to.

The field summary view first informs the analyst about the available fields in a given log file. The name of each field is provided as well as the number of distinct values this field exhibits. Finally, the bar chart displays the distribution of the values for this field. By taking up little space on the screen, it allows the analyst to easily notice and compare unusual distributions.

Each chart reacts to the current time filter applied in the time view panel and updates accordingly. An analyst can therefore inspect the evolution of the distributions for a given field by brushing and sliding the selection in the time view panel.

This feature is very effective for detecting massive events happening on a very short period such as brute-force attacks, DDoS, etc. In this kind of situation, the distributions of the values in some of the fields change noticeably in a very short period. For instance, in the case of a brute force attack against the admin password on a web service, a single or a few IP addresses will perform a noticeable share of the requests for a very short time and will therefore be overrepresented, but only for a few minutes. Depending on the size of the analysed log file and any potential sampling, the share of these requests could stay undetected. In contrast, when the analyst brushes the time view panel, (s)he can observe modifications at certain times in the IP address distributions which require further investigations.

To obtain more information about some fields, the analyst can click on summary representations to trigger the display of full-sized charts.

4.2.5 Full-Sized Charts View

The *full-sized charts view* contains the complete representations of the fields the analyst selected for exploration in the field summary view. A full-sized chart (see Fig. 4.2) exhibits the same colour as the log files it comes from. The values the field takes and the number of events with this values are provided.

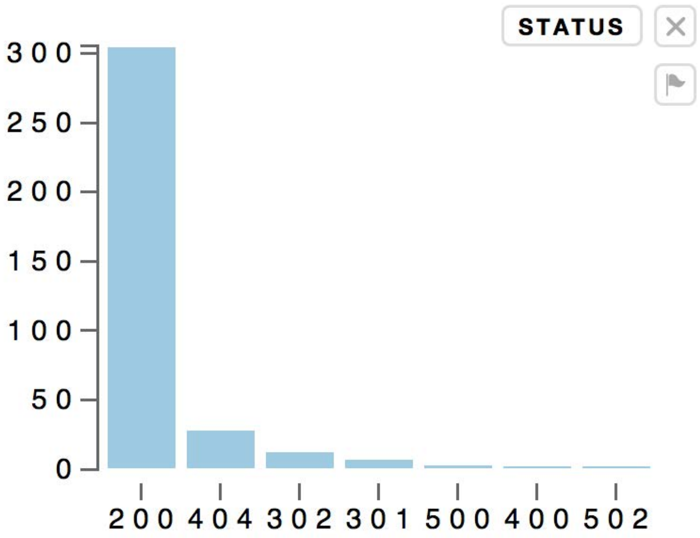


Figure 4.2: The full-sized chart of an HTTP Status code field.

CORGI is based partly on concepts already used in ELVis, such as generating visualisations based on the types of selected fields.

Representations are automatically selected according to the fields chosen by the analyst and using the same rules presented in ELVis (see p.65). For instance, Fig. 4.2 corresponds to the selection of a single `HTTP Status Code` which is *categorical* while Fig. 4.3 shows the representation produced for the selection of a `HTTP Status Code` field and an `IP` address field, both being *categorical* in nature. Pie charts were entirely replaced by bar charts, which are more effective for comparing and scaling to multiple values.

We should mention that CORGI only combines fields coming from the same log file into a single representation. Indeed, it would make no sense to combine different fields coming from different log files since their values cannot be safely related to shared single event.

Analysts can perform selections on the values of full-sized charts by clicking on them. The result of this selection is then applied to all the representations dealing with the same log file. For example, if the analyst selects the values `500` and `502` in the `HTTP Status Code` rep-

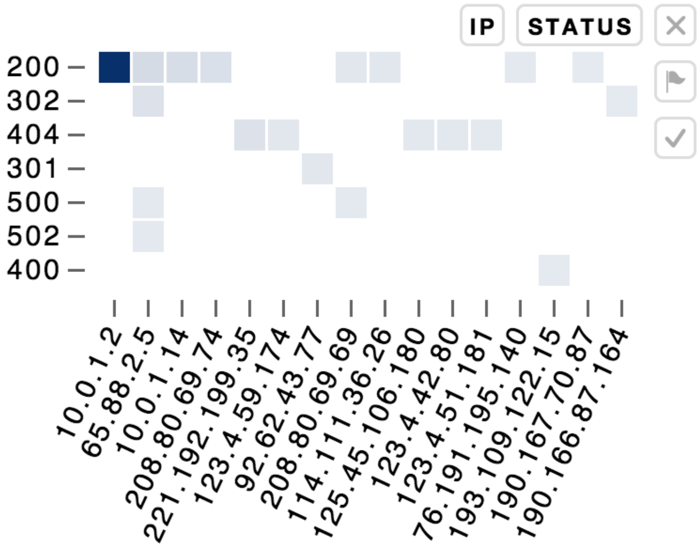


Figure 4.3: The full-sized chart of HTTP Status Code and IP fields.

resentation, all the representations for the same log file are modified to only display the events for which the `HTTP Status Code` is equal either to `500` or to `502`. For instance, the IP chart only displays IP addresses linked to requests which generated `500` or `502` `HTTP Status Code`.

Additional information and interactions are proposed on the upper right of each full-sized representation. First, the names of the represented fields are displayed. Three small buttons are available:

- The `cross` button removes the representation. The selections that were made using it are dismissed.
- The `check` button keeps this representation as is. If other fields that belong to the same log file are selected afterward in the same log file, they will not be added to this representation. It is however still possible to perform selections on it.
- The `flag` button collects values of interest.

When values of interest have been found in a chart, the analyst can click the `flag` button to collect these. For example, if the analyst selected `500` and `502` values in the `HTTP Status Code` and wants to keep the IP addresses which caused these values to be logged, (s)he can click on the `flag` button of the `IP` field representation. In the case of two dimensional visualisation such as a matrix chart, selecting a mark as a value of interest will produce two kinds of values of interest, one for each dimension. It is of course possible to generate values of interest for more than one field of the same log file.

In the next section, we show how values of interest are represented and how the analyst interacts with them.

4.2.6 Values of Interest Box

The *values of interest box* is located in the upper panel. It contains the values of interest which have been collected by the analyst. A given value of interest is represented as follows, left to right:

- The name of the field from which the values of interest were extracted, colour coded for the log file it came from.
- the semantic type of that field.
- the values of interest, which can be individually toggled.
- the names of the fields the values of interest can be applied to as filters, colour coded for the datasets they belong to.

Here, Fig. 4.4 shows a set of two values of interest obtained from the IP field in the blue dataset (in this case, an `apache access` log file), with the IP semantic type. It contains two values (`208.80.69.69` and `65.88.2.5`) and can be applied to the field named `IP` in the green log file. This last box is filled, which indicates that the filter is currently applied, but only for the currently selected address, which is `208.80.69.69`.

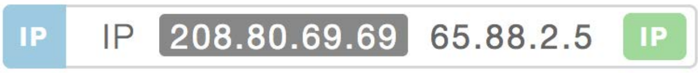


Figure 4.4: Two IP addresses selected as values of interest.

In order to remember from where these values of interest originate, the analyst can hover over the value of interest. A tool tip then displays the selections that led to obtain these values (in this case, `HTTP Status Code` = `500` or `502`).

Because the values of interest box is located in the upper panel, it allows the analyst to keep an eye on these values at all times. This helps to guide the selections that can be performed based on the values in the various fields of the displayed log files.

After this presentation of the interface and interactions used by CORGI, we provide details on their implementation and cases studies in the next section.

4.3 IMPLEMENTATION

CORGI is implemented as a web application using HTML5, Javascript, CSS and SVG. It uses React [57] to manage the user interface and is built following the Flux application architecture [58].

For effective visual exploration of logs, we consider certain functions essential such as filtering, annotation and support for multiple datasets. The user interface for CORGI relies heavily on reactive filtering and synchronisation between elements of the interface. Visualisation tools often accompany a chart with a similar smaller chart for context. PortVis [49] displays multiple synchronised time charts to allow for simultaneous global and detailed visualisation, as do Muelder *et al.* [51]. The time view in CORGI uses the same approach with two linked time charts, one global and one filtered, in an effort to help to provide both context and detail in one component simultaneously for multiple datasets.

When managing an interface composed of many linked components synchronised with shared data stores, maintaining a clean application state is difficult. One of the fundamental principles of React and Flux is to build applications around a unidirectional flow of data. The Flux architecture was proposed by Facebook as alternative to MVC style architectures for simplifying the task of managing an application with multiple synchronised views and data stores (Fig 4.5).

Using this foundation helped us build CORGI closer to a data pipeline model [22], [78].

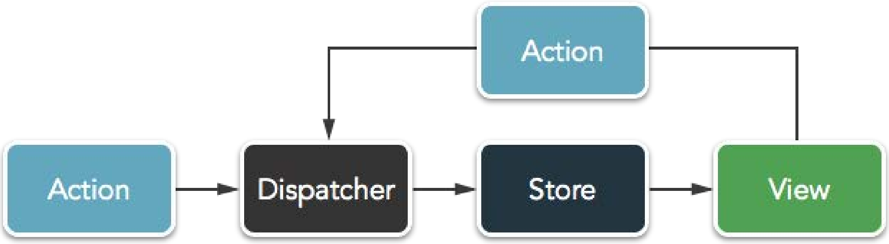


Figure 4.5: The Flux architecture implements a unidirectional flow of information. Data stores provide state for views, through which the user can trigger actions. These are dispatched to the relevant stores to change the state. These actions can also be issued by external sources, such as local or remote services.

Visualisations are generated using both D3 [10] and React. As both of these tools were designed to directly manipulate the DOM[12] but approach this in different ways, control has to be handed from React to D3 at some point. In our case, most of the SVG markup for our visualisations is generated by React, using D3 directly for axes and paths but also to manage layouts and scales.

Fast filtering is implemented using the Crossfilter library [13] which provides an OLAP server for interfacing with datasets as n-dimensional data cubes. As CORGI is entirely a client application with no server counterpart, information storage is limited but performed in memory and therefore much faster to access.

[12] The Document object model (DOM) is a convention for representing and interacting with objects through HTML5 documents

As it inherits its log parsing and chart selection capabilities from ELVis, CORGI can both parse multiple log formats and select charts automatically ((see listings 3 P.85 and 4 P.86).

```
Corgi.addParser(
  /(\S+) (\S+) (\S+) \[([^\]]+)\] "([^"]+)" (\S+) (\S+)(?:
 ↪  "([^"]+)")?(?: "(?:[^"]+)")?/,
  Corgi.mappings['apache-standard']
);
```

Listing 3: An Apache log file parser, which associates a regular expression with a field mapping (Fig 3.1, page 59).

```
Corgi.addMapping(’apache-standard’, [
  {
    index: 0,
    label: ’ip’,
    type: ’categorical’,
    semantics: ’ip’
  }, {
    index: 1,
    label: ’identd’,
    type: ’categorical’,
    semantics: ’id’
  }, {
    index: 2,
    label: ’userid’,
    type: ’categorical’,
    semantics: ’id’
  }, {
    index: 3,
    label: ’time’,
    type: ’time’,
    semantics: ’time’,
    transform: d3.time.format(’%d/%b/%Y:%X %Z’).parse
  }, {
    index: 4,
    label: ’request’,
    type: ’categorical’,
    semantics: ’HTTP Request’
  }, {
    index: 5,
    label: ’status’,
    type: ’categorical’,
    semantics: ’HTTP Status’
  }, {
    index: 6,
    label: ’size’,
    type: ’quantitative’,
    semantics: ’size’,
    unit: ’bytes’
  }, {
    index: 7,
    label: ’referer’,
    type: ’categorical’,
    semantics: ’URL’
  }, {
    index: 8,
    label: ’useragent’,
    type: ’categorical’,
    semantics: ’id’
  }
])
```

Listing 4: An Apache field mapping, which associates fields with the results of parsed log lines.

4.4 EXPERIMENTATION

We used CORGI to explore the 2012 HoneyNet visualisation Challenge [69] and the 2012 VAST Challenge [76] datasets.

The 2012 Honeynet visualisation challenge dataset contains log files that were gathered on a real compromised machine. In this dataset, there are about 10 different log files: auth.log, dpkg.log, kern.log, www-access.log, www-error.log, etc.

For this case study, we first inspected the www-access.log and www-media.log files, both logs from an apache web server, and the auth.log file, an authentication log for the host. The www-media.log file logs requests related to media such as images, CSS and Javascript files, while www-access.log logs main requests. The auth.log file contains different fields: timestamp, host, application, process id and the authentication message from which we can often extract an action, a user name and an IP address when the authentication is remote. Once these files were imported into CORGI, we first noticed inconsistencies between the www-media.log and www-access.log files thanks to the new layout and close proximity of the timelines. There is a burst of requests in the www-access.log file at a specific time while there are only a few requests at this same time in the www-media.log file. There are also two such bursts in the auth.log file, the latter apparently synchronised with one in the www-access.log file.

We first addressed the www-access.log file activity by inspecting the corresponding HTTP status codes, especially the server errors (5xx codes). We noticed that only two IP addresses are responsible for these errors (65.88.2.5 and 208.80.69.69), and marked these IP addresses accordingly, as *values of interest*. We then decided to track them through the three files using the *values of interest* box. In the www-access.log file, there are many other requests that did not seem harmful. Both the IP adresses are present in the www-media.log file. The second one generates several client errors by requesting a javascript URL not present on the server. By following these *values of interest* to the auth.log dataset, we noticed that there are eight SSH authentication successes and one failure linked to these IP addresses. For the second one, there are logged warnings of failed reverse DNS mappings. Two user names are used in these SSH connections: user1 and user3. These kinds of user names seemed strange on a host, although they could very well be due to the anonymisation process. We collected these as *values of interest* to further the exploration of the logs. Using CORGI, we were able to track activity for both these IP addresses across all three log files without copying filters or repeating the same actions for every file.

We had previously explored these log files using ELVis and found cascade like patterns indicating not only attempts at brute force remote access but also the coordination of these attempts from multiple sources. However, as we were unable to easily make links between the different log files, we were not able to easily use the other logs to obtain more information about the agents that generated this activity. CORGI can help collect and track *values of interest* across multiple logs. Thanks to these functions, we were able to look for requests in the Apache logs which generated errors and quickly look for the same adresses in our authentication logs.

For our second case study, we inspected log files from the 2012 VAST Challenge. The second mini-challenge is made of four log files. According to the described scenario, these logs come from the computer networks of a regional bank. Two files are 24-hour logs from a Snort IDS, while the others are firewall logs covering the same period. Due to performance constraints, we focused our analysis on the IDS log covering the first 24 hours.

Thanks to value based filtering, we discovered that 9 external IP addresses were the source of IRC traffic directed towards 314 different destination IP addresses in the internal network, and marked these 9 IP addresses as *values of interest* accordingly, referred to as A from now on. We also discovered 5 IP addresses in the internal network which are sources of scans of different services on the firewall. Again, we marked these 5 IP addresses as *values of interest*, referred to as B from now on. Our last discovery was more concerning: client hosts in the internal network scanned the firewall of the bank. We pursued our exploration by applying the IP addresses in B as a filter for the source IP address field. This allowed us to know whether they are the source of other alerts. As this is not the case, we then applied these *values of interest* as a filter for the destination IP addresses. This helped us to check whether these IP addresses were the target of attacks. This time we found that this is the case for 4 of the 5 IP addresses in B: these are implicated in the IRC traffic with the 9 source IP addresses in A. We therefore hypothesised that the hosts in B may have been part of a botnet controlled via IRC by the hosts in A.

During this use case analysis, we demonstrated how the ability to filter different field values, mark some as *values of interest* and use these values to filter the same dataset helps to conduct a coordinated exploration of various data files.

4.5 CONCLUSION

In this chapter we have presented CORGI, a web based tool designed to help explore multiple log files simultaneously and allow the user to traverse multiple datasets by following *points of interest* in the data. To do this reliably, fields are associated with `semantic types` which help discover related fields between log files. CORGI also builds upon the assistive visualisation capabilities of ELVis and extends several of its features to improve the exploration possibilities. The statistical type system used to match visualisations with selected data subsets is extended with a semantic system used for log traversal.

Fast filtering is now implemented for any view, and these views are now react synchronously across datasets following interaction with stored points of interest. The user interaction is an exploration guided by points of interest and an interface designed for search cycles with a drill-down approach. Not only can these points of interest be used to filter and link multiple logs, they also give an insight on the progression and results of an analysis, providing the essentials for sharing sessions and automatically generating reports.

CONCLUSION & FUTURE WORK

> "I love deadlines. I like the whooshing sound they make as they fly by."
>
> — Douglas Adams

We have discussed how visualisation offers a means for people to understand and manage data produced by information systems. Using visual representations of the current state and history of these systems, experts can analyse security information more intuitively and mitigate issues faster. Data pipelines can help transform the raw data from systems into data ready for visualisation. Parts of this data can then be strategically associated with *visual variables* which, when combined, give an understandable representation of this subset of information. In order to be well designed, visualisations are built by taking into account the particularities of human perception. They adhere to guidelines such as keeping data density high while restricting distractions to a minimum. The products of all these steps combined help security experts to better understand and interact with their data.

A study of visualisation tools produced by researchers helped us to understand how visualisation assists security operations in the field. Using scenarios, we identified problems and described personas which correspond to these problems. This let us define specialisations of security visualisations towards three complementary goals: monitoring, analysis and reporting.

The ELVis prototype was designed to simplify security analysis by contextually and automatically selecting appropriate visualisations for security experts. To help security experts who need to visualise security data but who lack training in visualisation, ELVis employs a design which separates visualisation knowledge from security knowledge. Visualisation experts add rules for ELVis to select relevant visualisations for specific data types. Security experts then describe their security datasources, which can then be matched with suitable visualisations. Demonstrating through experimentation that security practitioners could explore security information without prior training in visualisation confirmed the usefulness of ELVis and the validity of our approach. ELVis was published and presented at VizSec 2013 [33].

The CORGI prototype re-uses the concepts of its sibling and improves the process of exploring security data. CORGI was designed to provide an investigation experience and implements tools to help experts mark and collect *points of interest* in datasets. These can then be used to traverse and cross-reference multiple data sources in an analysis session without needing to leave the workspace. These *points of interest* also show promise for sharing sets of critical information and building reports. CORGI was published and presented at VizSec 2014 [32].

These two prototypes address the issues identified in security visualisation software: tools are often restricted to a single problem or data source, or require expertise in visualisation. Both ELVis and CORGI were designed to handle an extensible set of data formats and suggest relevant representations from an extensible set of visualisations. Thanks to these prototypes, we are closer now to designing tools which incorporate elements of monitoring, analysis and reporting in ways which let security experts work as close to their problems as possible while avoiding as many distracting intermediate hurdles as possible. Both of these prototypes have been tested against recognised security datasets, and have received positive feedback from security professionals.

These prototypes are offline web applications. They rely on local log files, a readily available source of security data, and are limited to the processing capabilities of a web browser. While unrelated to the problems addressed by both tools, combining this local approach with a more powerful remote backend would improve both log collection and data processing capacities thanks to scalable distributed storage and computing. This backend would allow the analysis of larger datasets spanning multiple systems using more powerful methods. As a shared resource, this would also open up possibilities for realtime updates, sharing sessions and even remote collaborative analysis.

To engineer a natural investigative experience, the design of CORGI is centred around collecting and linking evidence. One direct side effect of this approach is a new meta dataset, constructed by the investigator using human experience and containing meaningful information about the analysis itself. This additional dataset could be used to automatically build more meaningful reports which re-use the information about the path followed by the analyst during the investigation process. One of the goals for security visualisation tools is to help practitioners work together. With this combination of meaningful content and related context, collaborating team members could more easily share updates with each other, and third parties could be kept up to date thanks to reports generated automatically without sacrificing meaning. Furthermore, this dataset could also be used as context for future investigations, providing a reference point for visualisations and training material for automatic recommendations.

On a last note, this work is centred around people, a deciding factor in security. Security issues often focus on systems, and because we concentrate our efforts on solving software and hardware problems, we sometimes overlook people who are, after all, the end client. The most important objectives of security are to not only keep people safe, but also help them feel legitimately safe, in part through access to understandable information. Visualisation makes this last goal more accessible by humanising security information.

A CODE

Listing 5: The format used for parsing the VAST 2012 firewall logs.

```
Elvis.registerFormat({

  name: 'firewall log',

  pattern: /([^,]+),([^,]+),([^,]+),([^,]+),([^,]+),([^,]+),
   ↪  ([^,]+),([^,]+),([^,]+),([^,]+),([^,]+),([^,]+),
   ↪  ([^,]+),([^,]+),([^,]+)/,

  fields: [
    {
      label: 'time',
      type: ['time'],
      transform: d3.time.format('%d/%b/%Y %X').parse
    }, {
      label: 'syslog facility',
      type: ['categorical']
    }, {
      label: 'operation',
      type: ['categorical']
    }, {
      label: 'message code',
      type: ['categorical']
    }, {
      label: 'protocol',
      type: ['categorical']
    }, {
      label: 'source ip',
      type: ['categorical', 'ip']
    }, {
      label: 'destination ip',
      type: ['categorical', 'ip']
    }, {
      label: 'source hostname',
      type: ['categorical']
    }, {
      label: 'destination hostname',
      type: ['categorical']
    }, {
      label: 'source port',
      type: ['categorical']
    }, {
      label: 'destination port',
      type: ['categorical']
    }, {
      label: 'destination service',
      type: ['categorical']
    }, {
      label: 'direction',
      type: ['categorical']
    }, {
      label: 'connections built',
      type: ['categorical']
    }, {
      label: 'connections torn down',
      type: ['categorical']
    }
  ]

});
```

Listing 6: The format used for parsing the VAST 2012 CSV Snort logs.

```
Elvis.registerFormat({

  name: 'snort log',

  pattern: /([^,]+),([^,]+),([^,]+),([^,]+),([^,]+),([^,]+),
   ↪  ([^,]+),([^,]+),(?:(\S+) TTL:(\d+) TOS:(\S+) ID:(\S+)
   ↪  IpLen:(\d+) DgmLen:(\d+) (.*))?,(?:(\S+) Seq: (\S+)
   ↪  Ack: (\S+)  Win: (\S+)  TcpLen: (\d+))?,(.*)/,

  fields: [
    {
      label: 'time',
      type: ['time'],
      transform: d3.time.format('%m/%e/%Y %H:%M').parse
    }, {
      label: 'source ip',
      type: ['categorical', 'ip']
    }, {
      label: 'source port',
      type: ['categorical']
    }, {
      label: ’destination ip’,
      type: [’categorical’, ’ip’]
    }, {
      label: ’destination port’,
      type: [’categorical’]
    }, {
      label: ’classification’,
      type: [’categorical’]
    }, {
      label: ’priority’,
      type: [’categorical’]
    }, {
      label: ’label’,
      type: [’categorical’]
    }, {
      label: ’protocol’,
      type: [’categorical’]
    }, {
      label: ’ttl’,
      type: [’quantitative’]
    }, {
      label: ’TOS’,
      type: [’categorical’]
    }, {
      label: ’ID’,
      type: [’categorical’]
    }, {
      label: ’ip len’,
      type: [’quantitative’]
    }, {
      label: ’dgm len’,
      type: [’quantitative’]
    }, {
      label: ’packet info’,
      type: [’categorical’]
    }, {
      label: ’packet info 2’,
      type: [’categorical’]
    }, {
      label: ’sequence number’,
      type: [’categorical’]
    }, {
      label: ’ack’,
      type: [’categorical’]
    }, {
      label: ’windows’,
      type: [’categorical’]
    }, {
      label: ’tcp length’,
      type: [’quantitative’]
    }, {
      label: ’xref’,
      type: [’categorical’]
    }
  ]

});
```

ACRONYMS

IDS	Intrusion Detection Systems	2
NIDS	Network Intrusion Detection Systems	3
HIDS	Host Intrusion Detection Systems	3
AIDS	Application Intrusion Detection Systems	3
ELVis	Extensible log visualisation	
CORGI	Combination, Organisation and Reconstruction using Graphical Interactions	
DOM	Document object model	84
HTML	Hypertext markup language	
CSS	Cascading style sheets	68
SVG	Scalable vector graphics	68
D3	Data Driven Documents	

BIBLIOGRAPHY

[1] K. Abdullah and C. Lee, 'IDS RainStorm: Visualizing IDS Alarms', in *Proceedings of VizSEC'05*, 2005, pp. 1–10 (cit. on pp. 27, 29, 57).

[2] A. D. Amico and K. Whitley, 'The Real Work of Computer Network Defense Analysts The Analysis Roles and Processes that Transform', pp. 19–37, (cit. on pp. 55, 74).

[3] A. Analysis, 'Analyst ' s Notebook 8 Increase the depth of intelligence for effective resource utilization.', (cit. on p. 42).

[4] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 1st ed. Wiley, 2001, ISBN: 0471389226. [Online]. Available: `http://www.citeulike.org/user/thomasmuh/article/116315` (cit. on p. 2).

[5] J. Bertin, *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, ISBN: 0299090604. [Online]. Available: `http://www.citeulike.org/user/MoritzStefaner/article/258824` (cit. on pp. 10, 11, 64, 65).

[6] D. M. Best, A. Endert and D. Kidwell, '7 key challenges for visualization in cyber network defense', in *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, ser. VizSec '14, Paris, France: ACM, 2014, pp. 33–40, ISBN: 978-1-4503-2826-5. DOI: `10.1145/2671491.2671497`. [Online]. Available: `http://doi.acm.org/10.1145/2671491.2671497` (cit. on p. 5).

[7] A. F. Blackwell, C. Britton, A. Cox, T. R. G. Green, C. Gurr, G. Kadoda, M. Kutar, M. J. Loomes, C. L. Nehaniv, M. Petre, C. Roast, C. Roe, A. Wong and R. M. Young, 'Cognitive Dimensions of Notations: Design tools for cognitive technology.', *Technology*, CT '01, vol. 2001, no. Lnai 2117, M. Beynon, C. Nehaniv and K. Dautenhahn, Eds., pp. 325–341, 2001. [Online]. Available: `http://discovery.ucl.ac.uk/103930/` (cit. on p. 19).

[8] A. Blackwell, 'Cognitive Dimensions of Notations: Understanding the Ergonomics of Diagram Use', *Diagrammatic Representation and Inference*, vol. 5223, pp. 5–8, 2008. [Online]. Available: `http://dx.doi.org/10.1007/978-3-540-87730-1_4` (cit. on p. 19).

[9] *Blare IDS Blare/Introduction*. [Online]. Available: `http://www.rennes.supelec.fr/blare/` (cit. on p. 3).

[10] M. Bostock, V. Ogievetsky and J. Heer, ‘D3: Data-Driven Documents.’, *IEEE transactions on visualization and computer graphics*, vol. 17, no. 12, pp. 2301–9, Dec. 2011, ISSN: 1941-0506. DOI: 10.1109/TVCG.2011.185. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/22034350 (cit. on pp. 68, 84).

[11] E. F. Codd, ‘A Relational Model of Data for Large Shared Data Banks’, *Commun. ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970, ISSN: 0001-0782. DOI: 10.1145/362384.362685. [Online]. Available: http://doi.acm.org/10.1145/362384.362685 (cit. on pp. 75, 76).

[12] J. B. Colombe and G. Stephens, ‘Statistical Profiling and Visualization for Detection of Malicious Insider Attacks on Computer Networks’, pp. 138–142, 2004 (cit. on p. 27).

[13] *Crossfilter. Fast Multidimensional Filtering for Coordinated Views, http://square.github.io/crossfilter/* (cit. on pp. 68, 84).

[14] M. Daniel, S. Bohn, A. Wynne and A. William, ‘Real-Time Visualization of Network Behaviors for Situational Awareness’, 2010 (cit. on pp. 36, 37).

[15] H. Debar, M. Dacier and A. Wespi, ‘Towards a Taxonomy of Intrusion-Detection Systems’, *Computer Networks: Special Issue on Computer Network Security*, vol. 31, no. 9, pp. 805–822, 1999, ISSN: 1389-1286. [Online]. Available: http://www.citeulike.org/user/missiongiraffe/article/99836 (cit. on p. 3).

[16] S. G. Eick, M. C. Nelson and J. D. Schmidt, ‘Graphical analysis of computer log files’, *Commun. ACM*, vol. 37, no. 12, pp. 50–56, Dec. 1994, ISSN: 0001-0782. DOI: 10.1145/198366.198378. [Online]. Available: http://doi.acm.org/10.1145/198366.198378 (cit. on pp. 45, 49, 55).

[17] S. Engle and S. Whalen, ‘Visualizing distributed memory computations with hive plots’, *Proceedings of the Ninth International Symposium on Visualization for Cyber Security - VizSec '12*, pp. 56–63, 2012. DOI: 10.1145/2379690.2379698. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2379690.2379698 (cit. on p. 23).

[18] *Executive Insight | Think Quarterly by Google*. [Online]. Available: http://www.thinkwithgoogle.co.uk/intl/en_uk/quarterly/data/executive-insight-guy-laurence-ceo-vodafone.html (cit. on p. 4).

[19] G. A. Fink, P. Muessig and C. North, ‘Visual Correlation of Host Processes and Network Traffic’, no. 0106, pp. 11–19, 2005 (cit. on pp. 45, 47).

[20] F. Fischer, F. Mansmann and D. A. Keim, 'Real-Time Visual Analytics for Event Data Streams', in *Proc. of the 2012 ACM Symposium on Applied Computing*, ser. SAC '12, ACM, 2012 (cit. on p. 79).

[21] S. Foresti and J. Agutter, 'VisAlert : From Idea to Product', pp. 159–174, (cit. on pp. 23, 33, 34, 52).

[22] B. Fry, *Visualizing Data: Exploring and Explaining Data with the Processing Environment*. O'Reilly Media, 2008, p. 384, ISBN: 0596514557 (cit. on pp. 8, 84).

[23] J. Glanfield, S. Brooks, T. Taylor, D. Paterson, C. Smith, C. Gates and J. Mchugh, 'OverFlow : An Overview Visualization for Network Analysis', pp. 11–19, 2009 (cit. on pp. 33, 35).

[24] E. Godefroy, E. Totel, M. Hurfin and F. Majorczyk, 'Automatic generation of correlation rules to detect complex attack scenarios', in *10th International Conference on Information Assurance and Security, IAS 2014, Okinawa, Japan, November 28-30, 2014*, 2014, pp. 23–28. DOI: 10.1109/ISIAS.2014.7064615. [Online]. Available: http://dx.doi.org/10.1109/ISIAS.2014.7064615 (cit. on p. 4).

[25] J. R. Goodall and W. G. Lutters, 'Preserving the Big Picture : Visual Network Traffic Analysis with TNV', pp. 47–54, (cit. on pp. 44, 46, 57).

[26] T. R. G. Green, 'Cognitive dimensions of notations', *Proceedings of the fifth conference of the British Computer Society, Human-Computer Interaction Specialist Group on People and computers V*, pp. 443–460, 1989 (cit. on p. 19).

[27] L. Harrison, A. Lu and W. Wang, 'Interactive Detection of Network Anomalies via Coordinated Multiple Views', 2010 (cit. on pp. 42, 44).

[28] C. G. Healey, 'Perception in Visualization', 2006 (cit. on p. 13).

[29] C. G. Healey, K. S. Booth and J. T. Enns, 'High-speed visual estimation using preattentive processing', *ACM Transactions on Computer-Human Interaction*, vol. 3, no. 2, pp. 107–135, Jun. 1996, ISSN: 10730516. DOI: 10.1145/230562.230563. [Online]. Available: http://portal.acm.org/citation.cfm?doid=230562.230563 (cit. on p. 14).

[30] J. Heer, N. Kong and M. Agrawala, 'Sizing the Horizon : The Effects of Chart Size and Layering on the Graphical Perception of Time Series Visualizations', (cit. on p. 79).

[31] *Hive Plots - Linear Layout for Network Visualization - Visually Interpreting Network Structure and Content Made Possible*. [Online]. Available: http://www.hiveplot.net/ (cit. on p. 23).

[32] C. Humphries, N. Prigent, C. Bidan and F. Majorczyk, 'Corgi: combination, organization and reconstruction through graphical interactions', in *Proceedings of the Eleventh Workshop on Visualization for Cyber Security*, ser. VizSec '14, Paris, France: ACM, 2014, pp. 57–64, ISBN: 978-1-4503-2826-5. DOI: 10.1145/2671491.2671494. [Online]. Available: http://doi.acm.org/10.1145/2671491.2671494 (cit. on pp. 6, 92).

[33] C. Humphries, N. Prigent, C. Bidan and F. Majorczyk, 'ELVIS: Extensible Log VISualization.', in *Visualization for Cyber Security, VizSec '13, Atlanta, GA, USA, October 14, 2013*, 2013, pp. 9–16. DOI: 10.1145/2517957.2517959. [Online]. Available: http://doi.acm.org/10.1145/2517957.2517959 (cit. on pp. 5, 91).

[34] 'Inappropriate Content Visualization – Mark II | fifth.sentinel on WordPress.com', Tech. Rep. [Online]. Available: http://5thsentinel.wordpress.com/2009/10/19/inappropriate-content-visualization-mark-ii/ (cit. on pp. 23, 24).

[35] D. Inoue, 'DAEDALUS-VIZ : Novel Real-time 3D Visualization for Darknet Monitoring-based Alert System', pp. 72–79, 2012 (cit. on pp. 38, 40).

[36] B. Irwin and J. V. Riel, 'Using InetVis to Evaluate Snort and Bro Scan', (cit. on p. 32).

[37] C. Kintzel, J. Fuchs and F. Mansmann, 'Monitoring Large IP Spaces with ClockView', (cit. on pp. 37, 39, 56).

[38] H. Koike, 'Visualizing Cyber Attacks using IP Matrix', pp. 91–98, 2005 (cit. on pp. 27, 30, 31).

[39] H. Koike and K. Ohno, 'SnortView : Visualization System of Snort Logs', pp. 143–147, 2004 (cit. on p. 55).

[40] K. Lakkaraju, E. S. Ave and A. J. Lee, 'NVisionIP : NetFlow Visualizations of System State for Security Situational Awareness', pp. 65–72, (cit. on pp. 42, 43).

[41] K. Lakkaraju, A. Slagell, W. Yurcik and S. North, 'Closing-the-Loop in NVisionIP : Integrating Discovery and Search in Security Visualizations', pp. 75–82, 2005 (cit. on p. 52).

[42] C. P. Lee and J. A. Copeland, 'FlowTag : A Collaborative Attack-Analysis , Reporting , and Sharing Tool for Security Researchers', pp. 103–107, 2006 (cit. on pp. 51, 52).

[43] H. C. Lee, T. Palmbach, M. T. Miller and C. Y. Lee, *Henry Lee's crime scene handbook*. Business Weekly publications, 2003, ISBN: 9789867747976. [Online]. Available: http://books.google.fr/books?id=rJh7nQEACAAJ (cit. on p. 73).

[44] W. Lian, F. Monrose and J. Mchugh, 'Traffic Classification Using Visual Motifs : An Empirical Evaluation', 2010 (cit. on pp. 30, 31).

[45] Z. Liu and J. Heer, 'The effects of interactive latency on exploratory visual analysis', 2014 (cit. on p. 69).

[46] Y. Livnat, J. Agutter, S. Moon and S. Foresti, 'Visual correlation for situational awareness', *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, vol. 1, pp. 95–102, DOI: 10.1109/INFVIS.2005.1532134. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1532134 (cit. on pp. 23, 33, 34, 52).

[47] Y. Livnat, J. Agutter, S. Moon, R. F. Erbacher and S. Foresti, 'A Visualization Paradigm for Network Intrusion Detection', no. June, pp. 17–19, 2005 (cit. on pp. 33, 34, 52).

[48] J. Mackinlay, 'Automating the design of graphical presentations of relational information', *ACM Transactions on Graphics*, vol. 5, no. 2, pp. 110–141, Apr. 1986, ISSN: 07300301. DOI: 10.1145/22949.22950. [Online]. Available: http://www.citeulike.org/user/Yanno/article/989920 (cit. on pp. 10, 12).

[49] J. Mcpherson, P. Krystosk and L. Livermore, 'PortVis : A Tool for Port-Based Detection of Security Events', pp. 73–81, 2004 (cit. on pp. 27, 28, 30, 31, 84).

[50] A. Mordvintsev, C. Olah and M. Tyka, 'Inceptionism: going deeper into neural networks', Technical report, Google Inc., 2015. Google Research Blog, bit. ly/1BkXP09, Tech. Rep., 2015 (cit. on p. 15).

[51] C. Muelder, 'A Visualization Methodology for Characterization of Network Scans Workshop on Visualization for Computer Security', pp. 29–38, 2005 (cit. on p. 84).

[52] D. Norman, 'The design of everyday things. 1990', *Currency and Doubleday, New York*, (cit. on p. 17).

[53] A. Oliner, A. Ganapathi and W. Xu, 'Advances and challenges in log analysis', *Communications of the ACM*, vol. 55, no. 2, pp. 55–61, 2012, ISSN: 00010782. DOI: 10.1145/2076450.2076466 (cit. on p. 55).

[54] J. Pearlman and P. Rheingans, 'Visualizing Network Security Events Using Compound Glyphs from a Service-Oriented', (cit. on pp. 37, 40).

[55] S. Ranjan, 'BGP Eye : A New Visualization Tool for Real-time Detection and Analysis of BGP Anomalies', pp. 81–90, 2003 (cit. on pp. 42, 45).

[56] J. Rasmussen, K. Ehrlich, S. Ross, S. Kirk, D. Gruen and J. Patterson, 'Nimble Cybersecurity Incident Management through Visualization and Defensible Recommendations', 2010 (cit. on p. 34).

[57] *React - A JavaScript library for building user interfaces, http://facebook.github.io/react/* (cit. on p. 83).

[58] *React | Flux Application Architecture, http://facebook.github.io/react/docs/flux-overview.html* (cit. on p. 83).

[59] *Realsecure server sensor*. [Online]. Available: `http://www-935.ibm.com/services/in/en/it-services/realsecure-server-sensor.html` (cit. on p. 27).

[60] F. Roveta and P. Milano, 'BURN : Baring Unknown Rogue Networks', (cit. on pp. 46, 50).

[61] B. Schwartz, *The Paradox of Choice: Why More Is Less*. Ecco, 2003, p. 288, ISBN: 0060005688 (cit. on p. 54).

[62] E. Segel and J. Heer, 'Narrative visualization: telling stories with data.', *IEEE transactions on visualization and computer graphics*, vol. 16, no. 6, pp. 1139–48, 2010, ISSN: 1077-2626. DOI: `10.1109/TVCG.2010.179`. [Online]. Available: `http://www.ncbi.nlm.nih.gov/pubmed/20975152` (cit. on p. 9).

[63] B. Shneiderman, 'The Eyes Have It : A Task by Data Type Taxonomy The Eyes Have It : A Task by Data Type Taxonomy for Information Visualizations', 1996 (cit. on p. 9).

[64] *SIDAN's webpage*. [Online]. Available: `http://www.rennes.supelec.fr/ren/rd/cidre/tools/sidan/` (cit. on p. 3).

[65] 'Snort', Tech. Rep., Jan. 2013 (cit. on p. 3).

[66] J. Stoll, D. Mccolgin, M. Gregory, V. Crow and W. K. Edwards, 'Adapting Personas for Use in Security Visualization Design', (cit. on p. 24).

[67] T. Taylor, S. Brooks, J. Mchugh, P. By and J. Work, 'NetBytes Viewer : An Entity-based Netflow Visualization Utility for Identifying Intrusive Behavior', pp. 101–114, (cit. on p. 56).

[68] T. Taylor, D. Paterson, J. Glanfield, C. Gates, S. Brooks and J. McHugh, 'FloVis: Flow Visualization System', *2009 Cybersecurity Applications Technology Conference for Homeland Security*, pp. 186–198, 2009. DOI: `10.1109/CATCH.2009.18`. [Online]. Available: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4804443` (cit. on p. 56).

[69] The Honeynet Project, *Forensic Challenge 10 - "Attack Visualization" | The Honeynet Project, http://www.honeynet.org/node/781* (cit. on pp. 56, 68, 87).

[70] The Miso Project, *The Miso Project :: d3.chart, http://misoproject.com/d3-chart/* (cit. on p. 68).

[71] S. Tricaud, 'Picviz: Finding a Needle in a Haystack.', in *Proceedings of the First UNSENIX Workshop on the Analysis of System Logs (WASL)*, 2008 (cit. on pp. 45, 48, 55, 65).

[72] E. R. Tufte, *Beautiful Evidence*. Graphis Pr, 2006, ISBN: 0961392177. [Online]. Available: http://www.citeulike.org/user/anovstrup/article/7296772 (cit. on pp. 5, 17, 36).

[73] ——, 'Envisioning Information', *Taxon*, vol. 40, no. 1, p. 159, Feb. 1991, ISSN: 00400262. DOI: 10.2307/1222963. [Online]. Available: http://www.jstor.org/stable/1222963?origin=crossref%20http://www.citeulike.org/user/cmalek/article/227115 (cit. on p. 17).

[74] ——, *The Visual Display of Quantitative Information*. Graphics Press, 2001, p. 200, ISBN: 0961392142 (cit. on pp. 17, 18, 56).

[75] ——, *Visual Explanations: Images and Quantities, Evidence and Narrative*. Graphics Press, 1997, p. 156, ISBN: 0961392126. [Online]. Available: http://www.sudoc.abes.fr/DB=2.1/SRCH?IKT=12&TRM=008134480 (cit. on pp. 17, 55, 56).

[76] Visual Analytics Community. (2012). VAST Challenge 2012, [Online]. Available: http://www.vacommunity.org/VAST+Challenge+2012 (cit. on p. 87).

[77] C. Wagner, A. Dulaunoy, S. A. Ses and T. Engel, 'PeekKernelFlows : Peeking into IP flows', pp. 1–5, 2010 (cit. on p. 56).

[78] L. Wilkinson, *The Grammar of Graphics*. Springer, 1999, p. 408, ISBN: 0-387-98774-6. [Online]. Available: http://www.sudoc.abes.fr/DB=2.1/SRCH?IKT=12&TRM=071620559 (cit. on pp. 7, 8, 63–65, 84).

[79] C. V. Wright, F. Monrose and G. M. Masson, 'Using Visual Motifs to Classify Encrypted Traffic', pp. 41–50, 2003 (cit. on pp. 30, 31).

[80] T. H. Yu, B. W. Fuller, J. H. Bannick, L. M. Rossey and R. K. Cunningham, 'Integrated Environment Management for Information Operations Testbeds', pp. 67–83, (cit. on pp. 37, 38).

VU :

Le Directeur de Thèse
(Nom et Prénom)

VU :

Le Responsable de l'École Doctorale

VU pour autorisation de soutenance

Rennes, le

Le Président de l'Université de Rennes 1

Guy CATHELINEAU

VU après soutenance pour autorisation de publication :

Le Président de Jury,
(Nom et Prénom)