CORGI: Combination, Organization and Reconstruction through Graphical Interactions

Christopher Humphries, Nicolas Prigent, Christophe Bidan, Frédéric Majorczyk

CORGI

Combination, Organization and Reconstruction through Graphical Interactions

Christopher Humphries
Supélec/INRIA
first.last@inria.fr

Nicolas Prigent
Supélec
first.last@supelec.fr

Christophe Bidan
Supélec
first.last@supelec.fr

Frédéric Majorczyk
DGA-MI/Supélec
first.last@intradef.gouv.fr

ABSTRACT

In this article, we present CORGI, a security-oriented log visualization tool that allows security experts to visually explore and link numerous types of log files through relevant representations and global filtering. The analyst can mark values as *values of interest* and then use these values to pursue the exploration in other log files, allowing him to better understand events and reconstruct attack scenarios. We present the user interface and interactions that ensure these capabilities and provide two use cases based on challenges from VAST and from the Honeynet project.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General – Security and Protection; I.3.6 [**Methodologies and Techniques**]: Interaction Techniques; H.5.2 [**Information Interfaces and Presentation**]: [User Interfaces]

General Terms

Security, Visualization

Keywords

Visualization, Intrusion Detection, Forensics

1. INTRODUCTION

In the context of IT forensics, analysts rely on log files that record and describe what happens in various parts of the system (network segments, routers, servers, hosts, etc.). Log files exist in different formats and contain different information depending on the software that generated them and what the file describes. While they are invaluable sources of information, log files are often impractical to analyze both because of their human-oriented formats and because of their size (many gigabytes being not uncommon).

VizSec '14 November 10 2014, Paris, France

ACM 978-1-4503-2826-5/14/11 ...$15.00
http://dx.doi.org/10.1145/2671491.2671494

Many visualization tools (among which [8, 16, 25, 23]) 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.

Furthermore, security-oriented visualization tools often aim to explain global trends or detect events that are symptomatic of a specific attack. 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 [20]: once the analyst has found an interesting event, he or she must be able to discover any other related events, even if these events are stored 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.

In this paper, we present CORGI (Combination, Organization and Reconstruction using Graphical Interactions), a web-based visualization tool that allows analysts to import various types of log files and to visualize their contents. Many log file formats can be exploited, and new ones can easily be added using a straightforward and consistant description of log file formats. CORGI also implements an iterative process inspired by those used for IT forensics by allowing the analyst to use the *values of interest* he or she has found in a given dataset to filter events in other datasets, thus implicitly relating these datasets.

This paper is organized as follows. First, we explain how CORGI manages the various log files. Then, we detail how relations can be efficiently created between log files based on *values of interest*. In section 3, we describe the representations and user interactions in CORGI. In section 4, we provide some information about implementation and experimentations. Finally, we discuss related work in section 5.

2. LOG FILES

CORGI uses log files as data sources. In this section, we first present how log files are organized. 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 relations can be made between *a priori* unrelated datasets.

2.1 Log File Organization

While exhibiting various contents and formats, log files are generally organized 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*. In CORGI, we reasonably hypothesize that all the entries in a given log file contain the same fields, which can of course assume different *values*. 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. We should mention that a field can be assigned a default value if this field is not relevant to the entry. For instance, the `apache standard` log format uses the value '-' as a default placeholder.

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`. Figure 1 shows two lines from an `apache-access` log file extracted from a dataset made available by the Honeynet Project [2].

```
10.0.1.2 - - [19/Apr/2010:10:54:12 -0700] \
  "GET /feed/ HTTP/1.1" \
  200 16605 "-" "Apple-PubSub/65.12.1"
10.0.1.14 - - [19/Apr/2010:11:16:59 -0700] \
  "POST /wp-cron.php?doing_wp_cron HTTP/1.0" \
  200 - "-" "WordPress/2.9.2; http://www.domain.org"
```

Figure 1: Two `apache standard` log entries

This example contains two lines (and therefore two events). Every event in this log file is described with nine fields, each with a semantic meaning, in the following order: the `IP address` of the client which made the request to the web server, the `identd` of the client machine, the `userid` of the user if authenticated, the `timestamp` defining when the event occured, the first line of the `HTTP request` performed by the client, the `HTTP status code` returned by the server, the `size` of the answer, the client `referrer` and `user agent`.

In consequence, a *type* can be associated with each field. Types in CORGI are *semantic types*, in contrast with [14] for instance: 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.

In addition, a flag is associated with each type stating whether this type is *categorical* or not. A *categorical type* is a type for which identical values refer to the same meaning while different values refer to different meaning. For instance, the `CVE_ID` type is categorical: Two log entries having the same value for a `CVE_ID` field refer to the same vulnerability, while different `CVE_ID` values refer to different vulnerabilities. Similarly, identical values for an `IP Address` refer to the same network device, local addressing and NAT notwithstanding. In contrast, `size` and `CPU usage` for instance are not categorical: identical values may refer to different realities while similar events may be logged with different `CPU usage` and `size` values.

CORGI uses *semantic types*, and more specifically *categorical types* to create relations between datasets. This aspect will be described in section 2.3.

2.2 Operations on a Single Log File Contents

When he or she explores 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. During experiments we performed with a previous tool [14], we observed that analysts have different approaches for exploring the same log file. For instance, they do not start by exploring the same fields. These differences can be explained by the attacks the analysts have been exposed to recently as well as the information they have 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 [7], 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*, he or she 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 type of the field.

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

2.3 Relating Datasets

A given log file offers a local view of events on a specific part of the monitored system. As such, it allows 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.

2.3.1 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 for 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 indentified 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. Only elements considered of interest in one log file can be related to other log files by using them as filters. With this approach, we avoid combinatorial explosions which can occur when using *natural joins*[1] between log files. 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 he or she is better informed on analysis context.

2.3.2 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 by their time of occurrence. The knowledge that two events are simultaneous or that one happened after the other is particularly relevant for an analyst.

[1]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.

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, in busy systems, the produced datasets may be big. Finally, 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 Section 3, 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 next section, we present how these concepts are implemented in CORGI in terms of representations and user interactions.

3. VISUALIZATION AND USER INTERACTION

In this section, we describe the interface and the user interactions we designed to help explore logs using visualization tools. We first provide an overview of the interface, then some details about its different areas and how the analyst interacts with them.

3.1 Overview

The main interface in CORGI (see Fig. 2) is divided into 4 panels, each of which has a specific purpose linked to a step in the exploration process. We will 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 *timeview panel* which displays the event time distribution for each log file. When log files are selected in the timeview panel, their fields appear in the adjacent *fields summary view*, which displays field distributions using sparkline type bar charts. When more than one dataset is selected in the timeview panel, the fields of every selected dataset are displayed. The *full-sized chart view* is the main panel. It contains all the full-sized charts with axes and labels. Finally, the *values of interest box* 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 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. He or she can then obtain more information about the value distribution for each field and compare these across log files. If some fields look particularly interesting, he or she may then explore them further

Figure 2: An overview of CORGI.

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 prevents the analyst from being lost during the exploration process and also removes any constraint on the order in which the log files and fields are explored.

CORGI's interface also follows one following key unifying rule: each log file is assigned a color which is applied to its visualizations in the entire interface 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.

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

3.2 Importing Logs

CORGI uses a slightly modified version of the importation mechanism designed for [14]. A variety of log formats are available, among which `apache standard`, `syslog` and its variations, `Snort` alert logs and the `2012 VAST Challenge firewall log` [26]. Adding new log file formats is straightforward: each log file type is handled by a specific parser which provides a regular expression matching valid log entries for this format, as well as the name and type for each field. In consequence, if new types of log file need to be imported, this requires nothing more than the right regular expression, with the names and *semantic type* for each field, as well as an indication of whether or not the field is *categorical*. The need for a *semantic type* for each field and an indication of whether they are *categorical* or not 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 normalized[2], each field is mapped to its descriptors, the events are counted and the log time period is retrieved.

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

3.3 Timeview Panel

The *timeview panel* displays the distribution of events across time for the imported log files. It is composed of two similar representations which both visualize 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 [12]. 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 each log file. In Fig. 2 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 roll-overs 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 log-

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

ging 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 [9]. 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: synchronized attacks over multiple systems, appearing 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, he or she 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. This part provides complementary information about each log file: 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 the analyst.

3.4 Fields Summary View

The *field summary view* contains a summary chart representation of all the fields selected in the timeview panel. In order to help the analyst, each field exhibits the same color 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 timeview 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 timeview 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 noticibly 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 over-represented, but only for a few minutes. Depending on the size of the analyzed log file and any eventual sampling, the share of these requests could stay undetected. In contrast, when the analyst brushes the timeview panel, he or she 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.

3.5 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. 3) exhibits the same color as the log files it comes from. The values the field takes and the number of events with this values are provided.

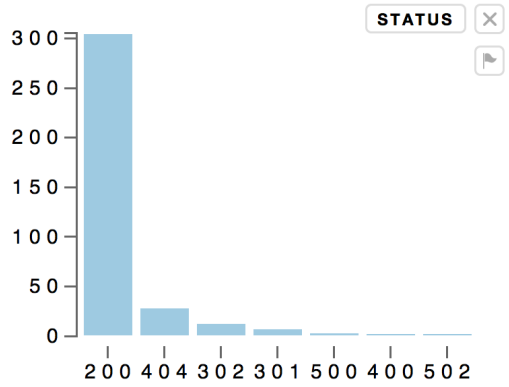

Figure 3: The full-sized chart of an HTTP Status code field.

Representations are automatically selected according to the field chosen by the analyst and using the same approach presented in [14]. For instance, Fig. 3 corresponds to the selection of a single `HTTP Status Code` which is *categorical* while Fig. 4 shows the representation produced for the selection of an `HTTP Status Code` and `IP` field, both being *categorical* in nature.

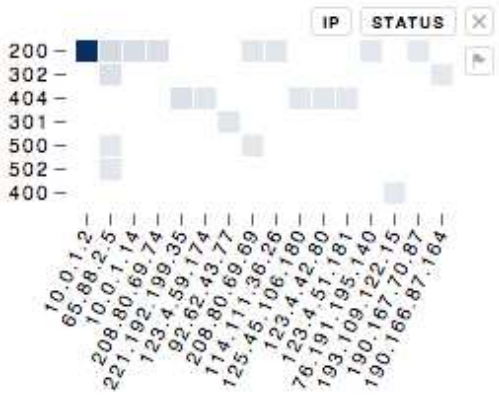

Figure 4: The full-sized chart of HTTP Status Code and IP fields.

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` representation, 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 `flag` button collects values of interest.
- 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.

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, he or she can click on the `flag` button of the `IP` field representation. 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.

3.6 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, color 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, color coded for the datasets they belong to.

Here, Fig. 5 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`.

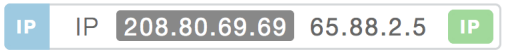

Figure 5: Two IP addresses selected as values of interest.

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. IMPLEMENTATION AND CASE STUDY

CORGI is implemented as a web tool using HTML5, Javascript, CSS and SVG. It uses React [3] to manage the user interface and is built upon the Flux application architecture [4]. These projects follow the fundamental premise that a unidirectional flow of data is central to the application. Using this foundation helped us build CORGI around data pipeline models [10, 27]. Although the SVG markup for our visualizations is generated using React, the d3.js library [6] is used to directly generate axes and paths as well as managing layouts and scales. Fast filtering is implemented using the Crossfilter library [1] which provides an OLAP server for interfacing with datasets as n-dimensional data cubes. As it inherits its log parsing and chart selection capabilities from ELVis [14], CORGI can both parse multiple log formats and select charts automatically with minimal effort.

We used CORGI to explore the 2012 HoneyNet Visualization Challenge [2] and the 2012 VAST Challenge [26] datasets on a MacBook Pro with 4GB of memory, a 2.5GHz Intel Core i5 processor and a GeForce GT 330M graphics card with 512MB of memory. We used Google Chrome version 36.

The 2012 Honeynet Visualization 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 will inspect the www-access.log and www-media.log files, which are logs from an apache web server, and the auth.log file, which is the authentication log for the host. The www-access.log and www-media.log files are consistent with the description given in Subsection 2.1. The latter logs the requests related to media such as images, CSS and Javascript files, while the first one logs the 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 the files are imported into CORGI, we first notice inconsistencies between the www-media.log and www-access.log files. There is a burst of requests in the www-access.log file at a specific time while there are only a few requests at the same time in the www-media.log file. There are also two bursts in the auth.log files, the latter seemingly synchronized with one in the www-access.log file.

We first address the www-access.log file activity by inspecting the HTTP status codes, especially the server errors (5xx codes). We notice that only two IP addresses are responsible for these errors (65.88.2.5 and 208.80.69.69), and mark these IP addresses accordingly, as *values of interest*. We then decide to track them in the three files using the *values of interest* box. In the www-access.log file, there are

many other requests that do 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 tracking these values of interest in the auth.log dataset, we notice that there were eight SSH authentication successes and one failure from both these IP addresses. For the second one, there are logged warnings of failed reverse DNS mappings. Two user names were used in these SSH connections: user1 and user3. These kinds of user names seem strange on a host, although they could very well be due to the anonymization process. We collect these as *values of interest* to continue the exploration of the logs. Using CORGI, we were able to easily track activity for both these IP addresses accross all three log files.

For our second case study, we inspected log files from the 2012 VAST Challenge. The second mini-challenge provides us with 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 space contraints, we cannot describe the full analysis of the VAST challenge and we will focus on part of our analysis of the IDS log covering the first 24 hours.

Thanks to the possibility of selecting and excluding values, 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, refered 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*, refered to as B from now on. Our last discovery is more concerning: client hosts in the internal network scanned the firewall of the bank. We pursue our exploration by applying the IP addresses in B as a filter for the source IP address field. This allows us to know whether they are the source of other alerts. As this is not the case, we then apply these *values* as a filter for the destination IP addresses. This allows us to know whether these IP addresses were the target of attacks. This time we find that this is the case for 4 of the 5 IP addresses in B: they are implicated in the IRC traffic with the 9 source IP addresses in A. We hypothesize that the hosts in B may be part of a botnet controlled via IRC by the hosts in A.

In this use case, we show how the ability to select and exclude different field values, mark some as *values of interest* and use these values to filter the same dataset can quickly and easily provide clues about what happened in log files.

5. RELATED WORKS

CORGI is based partly on concepts already used in ELVis, such as generating visualizations based on selected typed fields. This automatic selection is based on principles close to those used in Autovis [28]. We use the nature of the content, statistical analysis of the data aspects and rules from the Grammar of Graphics [27] to select and generate visualizations. ELVis was designed using the same principles but was optimized for security data. CORGI takes this logic and security tuned design further. For example, we assume that network events can all be correlated and compared in time and have designed a chart to better exploit that.

For effective visual exploration of logs we consider certain functions essential such as filtering, annotation and support for multiple datasets. As explained previously, the interface in CORGI relies heavily on reactive filtering and synchronization for elements of the interface. PortVis [21] displays multiple synchronized time charts to allow for both global and detailed visualization, and Muelder et al. use a global view synchronized with a detailed view [22]. 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.

To provide exploration using a drill-down approach, we rely on synchronized update and filtering in charts. NVisionIP [17] follows a similar approach using multiple charts to provide this drill-down interaction cycle as does ClockView [15] on a larger scale using four synchronized visualizations designed for successive and progressive filtering. Harrison et al. designed a tool [11] in which each visualization is designed to provide access to the different features of networks: spatial disposition using graphs, time histograms for network events and a scatterplot visualization for spectral data analysis. Tools linked to real world forensics such as the Analysts Notebook [5] also use combinations: maps for localizing forensic data and graphs to then visualize links in this data. We follow the same approach without a predefined set of visualizations, instead letting the operator decide to create visualizations best suited for that point of the exploration.

Increasing its similarity to CORGI, improvements to NVisionIP add the capacity for recording and reusing effective search patterns [18] based on features often found in certain anomalies. Further improvements also add sparkline charts [29] for increased data density.

Aiming to improve the reactivity and available options for corporate security operators, Hertzog [13] proposes building visualisations by first reducing the data to those members deemed most important, and then to group these events by meta-application, e.g. web applications. We reason that by comparing and relating datasets using their fields and values, we implicitely group data by importance and feature.

The two tools that seem closest to this project are Polaris [24] and FlowTag [19]. Polaris takes a similar approach to statistical typing, choosing to identify fields as either ordinal or quantitative, and then using these to assist the user in creating visualizations. These are produced by combining the fields of multiple sources of data into different layers of visualization. Filtering is made possible using a data cube which directly maps a database schema. The authors then describe the direct link between this progressive filtering and the progressive creation of relational queries to a database, and hypothesize that the tuples selected graphically and used for queries can themselves be exported as a further dataset. FlowTag adopts a less direct approach to extracting relevant information, and uses filtering to exclude irrelevant dataflows combined with tagging to identify and characterize relevant flows. This approach aims to improve the quality of reports and automatic report generation.

6. CONCLUSION AND FUTURE WORK

We have presented CORGI, a web based tool for the simultaneous exploration and semantic relating of multiple log files. CORGI is built upon the assistive visualization capabilites of ELVis, implements synchronized views, fast data cube type filtering and guided exploration using points of

interest. These values of interest can then be used to filter and link multiple logs.

We have many plans for extending this system. The current state of the tool relies on local files and the processing capabilities of browsers. Extending this model to a server backend for log collection and aggregation would let us explore larger files spanning multiple systems using more complex data operations, including real time updates. We are already experimenting with UI state serialization, and using a server backend would also open-up new possibilites for collaboration and sharing forensic analysis sessions.

Once values of interest are collected and verified they can be stored and exported as new datasets. In the long term one of our goals is to use these to help generate reports and improve future exploration assistance. As this information can be imported and then directly reused in CORGI these reports could themselves be analysed and related, essentially starting a new interactive session and using the same advantages provided during the initial exploration.

7. ACKNOWLEDGEMENTS

The authors would like to thank the DGA-MI for financing this research project.

8. REFERENCES

[1] Crossfilter. Fast Multidimensional Filtering for Coordinated Views, http://square.github.io/crossfilter/.

[2] Forensic Challenge 10 - "Attack Visualization" | The Honeynet Project, http://www.honeynet.org/node/781.

[3] React - A JavaScript library for building user interfaces, http://facebook.github.io/react/.

[4] React | Flux Application Architecture, http://facebook.github.io/react/docs/flux-overview.html.

[5] A. Analysis. Analyst ' s Notebook 8 Increase the depth of intelligence for effective resource utilization.

[6] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, Dec. 2011.

[7] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, June 1970.

[8] S. G. Eick, M. C. Nelson, and J. D. Schmidt. Graphical analysis of computer log files. *Commun. ACM*, 37(12):50–56, Dec. 1994.

[9] 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*, SAC '12. ACM, 2012.

[10] B. Fry. *Visualizing Data: Exploring and Explaining Data with the Processing Environment*. O'Reilly Media, 2008.

[11] L. Harrison, A. Lu, and W. Wang. Interactive Detection of Network Anomalies via Coordinated Multiple Views. In *Proc. of VizSEC'10*, pages 91–101, 2010.

[12] 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.

[13] P. Hertzog. Visualizations to Improve Reactivity Towards Security Incidents Inside Corporate Networks. In *Proc. of VizSEC'06*, pages 95–101, 2006.

[14] C. Humphries, N. Prigent, C. Bidan, and F. Majorczyk. Elvis: Extensible log visualization. In *Proc. of VizSEC'13*, pages 9–16. ACM, 2013.

[15] C. Kintzel, J. Fuchs, and F. Mansmann. Monitoring Large IP Spaces with ClockView. In *Proc. of VizSEC'11*, pages 2:1–2:10, 2011.

[16] H. Koike and K. Ohno. SnortView: Visualization System of Snort Logs. In *Proc. of VizSEC/DMSEC'04*, pages 143–147, 2004.

[17] K. Lakkaraju, E. S. Ave, and A. J. Lee. NVisionIP: NetFlow Visualizations of System State for Security Situational Awareness. In *Proc. of VizSEC/DMSEC'04*, pages 65–72, 2004.

[18] K. Lakkaraju, A. Slagell, W. Yurcik, and S. North. Closing-the-Loop in NVisionIP: Integrating Discovery and Search in Security Visualizations. In *Proc. of VizSEC'05*, pages 75–82, 2005.

[19] C. P. Lee and J. A. Copeland. FlowTag: A Collaborative Attack-Analysis, Reporting, and Sharing Tool for Security Researchers. In *Proc. of VizSEC'06*, pages 103–108, 2006.

[20] H. Lee, T. Palmbach, M. Miller, and C. Lee. *Henry Lee's crime scene handbook*. Business Weekly publications, 2003.

[21] J. Mcpherson, P. Krystosk, and L. Livermore. PortVis: A Tool for Port-Based Detection of Security Events. In *Proc. of VizSEC/DMSEC'04*, pages 73–81, 2004.

[22] C. Muelder, K. L. Ma, and T. Bartoletti. A Visualization Methodology for Characterization of Network Scans Workshop on Visualization for Computer Security. In *Proc. of VizSEC'05*, pages 29–38, 2005.

[23] A. Oliner, A. Ganapathi, and W. Xu. Advances and challenges in log analysis. *Communications of the ACM*, 55(2):55–61, 2012.

[24] C. Stolte and P. Hanrahan. Polaris: a system for query, analysis and visualization of multi-dimensional relational databases. In *Proc. of the IEEE Symposium on Information Visualization*, pages 5–14, 2000.

[25] S. Tricaud. Picviz: Finding a Needle in a Haystack. In *Proc. of the First USENIX Workshop on the Analysis of System Logs (WASL)*, 2008.

[26] Visual Analytics Community. VAST Challenge 2012, 2012.

[27] L. Wilkinson. *The Grammar of Graphics*. Springer, 1999.

[28] G. Wills and L. Wilkinson. AutoVis: Automatic visualization. *Information Visualization*, 9(1):47–69, Dec. 2008.

[29] W. Yurcik. Tool Update: NVisionIP Improvements (Difference View, Sparklines, and Shapes). In *Proc. of VizSEC'06*, pages 65–66, 2006.