ICIMP 2015

The Tenth International Conference on Internet Monitoring and Protection

ISBN: 978-1-61208-413-8

June 21 - 26, 2015

Brussels, Belgium

ICIMP 2015 Editors

Constantion Paleologu, University ‘Politehnica’ Bucharest, Romania
Jaime Lloret Mauri, Polytechnic University of Valencia, Spain

ICIMP 2015

Foreword

The Tenth International Conference on Internet Monitoring and Protection (ICIMP 2014), held between June 21-26, 2015, in Brussels, Belgium, continued a series of special events targeting security, performance, vulnerabilities in Internet, as well as disaster prevention and recovery.

We take here the opportunity to warmly thank all the members of the ICIMP 2015 Technical Program Committee, as well as all of the reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to ICIMP 2015. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the ICIMP 2015 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that ICIMP 2015 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the field of Internet monitoring and protection.

We are convinced that the participants found the event useful and communications very open. We hope that Brussels, Belgium, provided a pleasant environment during the conference and everyone saved some time to enjoy the charm of the city.

ICIMP 2015 Chairs:

ICIMP Advisory Committee
Go Hasegawa, Osaka University, Japan
Sathiamoorthy Manoharan, University of Auckland, New Zealand
Jaime Lloret Mauri, Polytechnic University of Valencia, Spain
Constantion Paleologu, University ‘Politehnica’ Bucharest, Romania
Michael Grottke, University of Erlangen-Nuremberg, Germany
Emmanoil Serelis, University of Piraeus, Greece
William Dougherty, Secern Consulting - Charlotte, USA

ICIMP Industry/Research Chairs
Matthew Dunlop, United Sates Army Cyber Command, USA
Mohamed Eltoweissy, Pacific Northwest National Laboratory, USA
Nicolas Fischbach, COLT Telecom, Germany
Emir Halepovic, AT&T Labs - Research, USA
Miroslav Velev, Aries Design Automation, USA
Wei Wang, SnT Centre, University of Luxembourg, Luxembourg
Wenjing Wang, Attila Technologies, USA
Steffen Wendzel, Fraunhofer FKIE, Germany
Artsiom Yautsiukhin, National Council of Research, Italy

ICIMP 2015

COMMITTEE

Table of Contents

Visualization and Monitoring for the Identification and Analysis of DNS Issues

Christopher Amin, Massimo Candela, Daniel Karrenberg, Robert Kisteleki and Andreas Strikos
Réseaux IP Européens (RIPE) Network Coordination Centre
Amsterdam, Netherlands
Email: camin@ripe.net, mcandela@ripe.net, daniel.karrenberg@ripe.net, robert@ripe.net, astrikos@ripe.net

***Abstract*—The user experience of an Internet service depends partly on the availability and speed of the Domain Name System (DNS). DNS operators continually need to identify and solve problems that can be located at the end user, a name server, or somewhere in between. In this paper, we show how DNSMON, a production service for measuring and comparing the availability and responsiveness of key name servers, correlates and visualizes different types of measurements collected by RIPE Atlas vantage points worldwide. DNSMON offers an interactive view, both historic and near real-time, at different levels of detail. It has successfully revealed and allowed analysis of many operational issues, including less obvious ones.**

Keywords–DNSMON; DNS; monitoring; network visualization.

I. Introduction

The Internet Domain Name System (DNS) [1][2] provides a mapping from user-visible domain names to other identifiers, such as network layer addresses. The performance of most Internet services can be perceptibly influenced by the quality and responsiveness of the DNS. Since it is a distributed system, its performance depends in turn on a number of elements, such as the authoritative name servers, caching name servers, local resolvers and the network in between. Due to the importance of this infrastructure, there have been many monitoring projects collecting data about different aspects of DNS, such as stability, security, performance, and traffic. Some of these projects also provide visualizations for administrators and decision makers — especially the DNS operators at various levels.

The visualization of Internet measurements in general has seen a growing interest in recent years. This is mainly due to the huge amount of data collected by Internet measurements projects, which cannot be fully understood and explored without meaningful visual representations. There are various free and commercially available DNS visualization tools, some examples being: DNSViz [3], which, given a domain name, visualizes the chain of name servers and certificates involved in the DNSSEC chain of trust; Flying Term [4], which offers a visualization to help administrators identify and understand DNS querying behavior due to anomalies such as misconfiguration and security events; DNSential [5], which allows users to issue custom IP address or domain name queries returning a graphical depiction of IP/domain relationships over time; RTIVS [6], which helps administrators to display and detect DNS amplification attacks; and VisualK [7], a system that visualizes how and when the clients of K-root name server migrate from one instance to another.

DNSMON [8] is a monitoring project started in 2001 to actively measure authoritative DNS servers at the root and top-level domain (TLD) level, from a large enough number of vantage points to reliably identify issues at or close to the servers themselves. While the majority of the DNS tools, including the tools cited above, focus on data collected by a specific server or through DNS resolutions from the resolver point of view, DNSMON aims to constantly monitor all the name servers belonging to entire zones — considered strategic for the functioning of the whole Internet — through performance measurements. It was initially conceived as a response to claims that root name servers performed poorly. Such claims were often based on measurements from one or — at most — a handful of vantage points and thus heavily influenced by network performance on a small number of network paths. The first implementation of DNSMON was based on a small process running on the nodes of the RIPE Test Traffic Measurement (TTM) [9] network. This process executed DNS queries and reported response times to a central server which stored them in round-robin database (RRD) [10] files. Users were able to monitor name servers and DNS zones through several automatically generated, non-interactive images. Due to the limitations of RRD and the static images, it was not possible to view results from arbitrary time periods at arbitrary levels of detail, nor to interactively access related measurements.

The project has proven its usefulness amongst operators and has evolved over time. In this paper, we will present the latest stage of DNSMON. The system is now based on the RIPE Atlas [11] network measurement project, which currently counts around 8000 active probes and 100 anchors worldwide carrying out more than 2000 network measurements per second. An anchor is a high-end machine colocated in a professional data center with defined network requirements; a probe is a small device usually located at end user sites. We decided to use measurements performed by anchors rather than probes in the interest of providing more accurate, reliable and consistent information without needing to compensate for erratic uptimes. It is also beneficial to avoid possible noise present in end user networks, since the focus is to identify issues near to core services.

Although this tool targets mostly network operators, the services they operate are sufficiently important that by facilitating analysis and quality of service improvement to diverse locations around the world, significant benefits are conferred to all users of the wider Internet.

Figure 1 shows a snapshot of the DNSMON interface. The main objectives of the system are as follows. The user selects a DNS zone z and an interval of time $\mathcal{T}$, the system shows a visual overview of the results of the measurements performed by the anchors against the set $\mathcal{S}$ of name servers belongong to z over time. The servers in $\mathcal{S}$ are detected in advance by performing an Name Server (NS) lookup on the zone, and collecting all unique IP addresses found in referenced A or AAAA records. This type of visualization

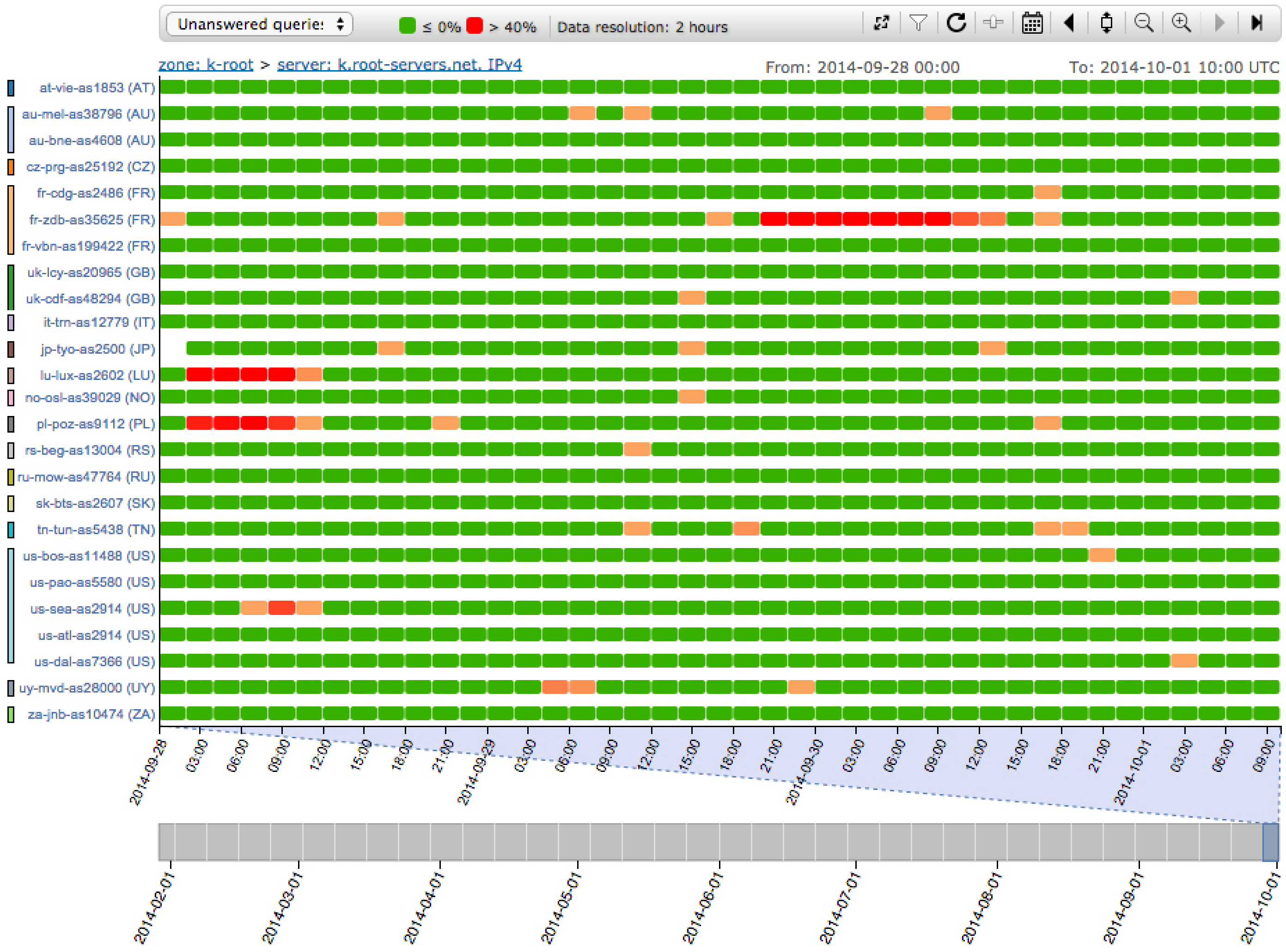


Figure 1. The main interface of DNSMON.

should be effective in giving a close to real-time overview of the quality of the entire zone at a glance and flexible and interactive enough to increase the details during analysis, while also correlating different types of measurement results and distinguishing abnormal from ordinary situations. The back end should only provide the data while the visualization and data correlation should be completely client-side, accessible from a web browser without third-party plug-ins. A typical use of our system is the following. Let γ be a geographical region in which the anchor α is deployed. Suppose that during $\mathcal{T}$ the DNS resolution of z exhibited some packet loss or high latency problem. Is there a specific name server s involved in the problem? Is the problem bound to γ? How can we exclude the possibility that the problem is related to a malfunction of α? Moreover, how did α reach s before and after the issue and what DNS responses were received?

The paper is organized as follows. In Section II, we describe the adopted visualization approach and introduce some formal terminology. Section III gives an overview of the general method for finding and solving problems using the tool, as well as some specific examples. In Section IV, we describe the collection and processing of measurement data used to support the visualization. In Section V, we outline the implementation of our tool and the technical challenges we faced. In Section VI, we present our conclusions and mention future directions.

II. Visualization Approach

For each monitored zone in DNSMON, we look up all the name servers defined for that zone, and schedule DNS measurements against them. The primary scheduled measurements are periodic Start of Authority (SOA) queries executed by a common subset of the RIPE Atlas anchors. Assumptions about the periodicity of the measurements are not hardcoded and may be varied in the future, depending on the monitoring needs.

As stated in the user requirements in Section I, the user should be able to monitor an arbitrary interval $\mathcal{T}$, up to the entire available history in the system. Therefore, the visualization should be able to represent both the potentially large number of results collected during $\mathcal{T}$ and to clearly convey the zone status without requiring interaction.

The visualization methodology we adopted is presented below together with supporting motivations. We discarded solutions based on line charts or time animations. Line charts are often used, in combination with downsampling algorithms, to represent trends in large amounts of data. The purpose of the downsampling is to try to retain the visual characteristics of the original line using considerably fewer data points, which in a web application results in a helpful reduction in the number of Document Object Model (DOM) elements to be handled by the browser [12]. At the same time, line charts require great attention to the axes, especially during comparisons of different trends. Moreover, we need to represent multiple trends concurrently and this requires too much space for an intelligible representation. A time animation, instead, can solve the space problem by representing a portion of $\mathcal{T}$ at a time, but requires the user to interact with the system and it does not provide an immediate overview of the whole $\mathcal{T}$, violating two base requirements.

We decided to follow Shneiderman's mantra of "overview first, zoom and filter, then details-on demand" [13] and to adopt a matrix with two axes. Each cell represents a measured value for the visualized zone z by using colors. Since the canvas space is limited and a single cell should be big enough to be clearly distinguishable and allow the user to interact with it, a data aggregation mechanism has been adopted representing groups of results instead of single values. $\mathcal{T}$ is represented on the x-axis and is divided into sub-intervals. Each cell represents an evaluation of the aggregation of all the results collected by the system in a certain sub-interval. The aggregation drastically reduces the number of represented elements, making it possible to use a performant and purely client-side browser visualization approach. We use the term "*native resolution*" for the case in which $\mathcal{T}$ is so small that each sub-interval contains a single measurement result.

In addition to z and $\mathcal{T}$, a name server s can be specified as an input. If s is specified, the set of anchors $\mathcal{A}$ monitoring z is represented on the y-axis. In this case, a cell refers to a measurement executed by a specific anchor and involving s at a sub-interval $t \in \mathcal{T}$. Instead, if s is not specified, all the name servers in $\mathcal{S}$ are represented on the y-axis. In this second case, a cell refers to the aggregation of all the data collected by all the anchors in $\mathcal{A}$ against a specific name server in $\mathcal{S}$ at a sub-interval $t \in \mathcal{T}$. By default s is not specified, a choice aligned with the aim of fostering incremental information enrichment, but it is possible to specify it by clicking on a name server label on the y-axis. The elements on the y-axis are grouped by using colored rectangles placed on the left of the axis labels. The colors are computed with the algorithm described in [14] to ensure that they are distinguishable from each other. The IP addresses, both IPv4 and IPv6, derived from lookups of the same NS record are visually close and grouped together, allowing a comparison between protocols and isolation of possible protocol-specific routing problems, while also satisfying modern operators who are as concerned with IPv6 reachability as IPv4. The anchors are grouped by the country code [15] where they are deployed, adding the possibility of seeing geographic correlations.

The user can select a *point of view*, or rather which qualitative aspect of the measurement to visualize. In particular, in the actual release, the available points of view are: packet loss, Round Trip Time (RTT), and relative RTT. Each cell in the *packet loss and RTT views* depicts respectively the percentage of packets lost and the amount of milliseconds of RTT measured for the sub-interval represented by the cell. The *relative RTT view* shows the percentage disparity of each sub-interval relative to the minimum measured in each row, highlighting unexpectedly high RTT values, and smoothing ordinarily high latencies. As a first approach we tried to represent more than one point of view at the same time, but this solution was discarded for the following reasons: it requires different graphical metaphors resulting in an increased complexity of the scene; given the amount of data to be represented, introducing additional elements into the scene causes cluttering and performance issues; and some users may be interested in only one qualitative aspect while, since the points of view are strongly related, spotting a problem in one will often indicate the same issue in another, causing an unnecessary information overload.

The cells are colored according to a threshold pair $c1, c2$. Cells corresponding to values below $c1$ are shown in green, above $c2$ in red, and between $c1$ and $c2$ in a color from a gradient domain between green and red. When a value is not available, the coresponding cell is shown in gray. In the measurements performed by DNSMON, lower values indicate better results. For this reason the choice of the colors reflects a transition from positive low values (green) to negative high values (red) [16]. The aim of the two thresholds is to create a visual separation between values considered good and bad, reducing the noise on the matrix. This is achieved by creating two separate ranges — the expected acceptable values, and the unacceptable ones — showing the gradient only for the cells between the two. A pair $c1, c2$ is defined for each point of view, by default tollerant threshold values are set. The user can easily tune $c1, c2$ to change the sensitivity of the tool and adapt it to the specific case. In addition to the coloring, it is possible to obtain the textual values of a cell, along with other information, by hovering over it with the mouse.

The user interaction plays a major role in our visualization. It is possible to zoom the viewport in and out in order to reduce or increase the visualized interval $\mathcal{T}$. When $\mathcal{T}$ changes, the data resolution also changes. In addition, $\mathcal{T}$ can be shifted in time, thereby maintaining the same data resolution. A time overview at the bottom shows the current extent of $\mathcal{T}$ within the total monitored period for that zone. The time can be easily navigated by interacting with the control panel, with the time overview, or simply by scrolling with the scroll wheel or by using the arrow keys on the matrix. By clicking on a cell it is possible to obtain different types of correlated measurement results. This feature is very effective during issue analysis. The reader can experiment with the current version of DNSMON by visiting the address https://dnsmon.ripe.net.

III. Discovering and Analyzing Abnormalities

The visualization makes it easy for the user to spot abnormal situations and, once discovered, to interactively provide access to extra information helpful for analysis and diagnosis. The first stage of the discovery is usually to look at the matrix and attempt to discern visual patterns in the coloring of the cells. Various patterns provide clues to the presence and nature of an abnormality. For instance, the existence of one or more horizontal red lines in the zone view may answer the question of whether there is a problem that involves a specific name server — if all but one row is green then we can probably determine that there is. Similarly, in the server view, red cells spanning multiple anchors in the same country may tell us that a problem is bound to a particular region, e.g., when a routing problem involving a particular internet exchange affects nearby anchors. When there is a problematic row corresponding to a single anchor, it is essential to understand whether that is a hint of a genuine network problem, or whether the anchor itself is malfunctioning. In this case, the user can opt to see the recent results from that anchor for every name server, as shown in Figure 2(d). If similar problems are seen for other name servers then it is likely that the anchor has some general issue. Also interesting are the edges before and after a visible period of disruption, as they may provide clues as to the cause of the problem and subsequent recovery. It is useful to request extra information for the cells here — in particular, comparing traceroutes side-by-side can show information about changes in packet routing behavior. Looking at the results of approximately concurrent HOSTNAME.BIND queries indicates which specific instance of a load-balanced or anycast service is answering DNS queries. The details for the SOA queries themselves allow more in-depth analysis, e.g., it may be possible to guess that a response has passed through an intermediate cache by looking for simplifications or other forms of answer mangling.

Operators and researches can use clues in the visualization and correlated data to determine the likely root causes, and share specific views that describe the exact details.

We present the following examples of real network events, which were spotted and analyzed thanks to our tool. Technical details and names have been intentionally removed from two of the examples to obscure identities of the affected services.

In our first case, depicted in Figure 2(a), a zone was involved in a major outage. The situation was immediately visible in DNSMON, where none of the anchors were successful in executing the measurements. The start and end times of the red areas are aligned perfectly across topologically and geographically distinct anchors, which points out that all the servers were affected at the same time suggesting a global service cause. Notice the orange borders around the main red area are due to the data aggregation, where positive and negative responses fall within the same sub-interval. Further zooming in revealed the exact start and end times of the event, with a sharp transition between green cells, without packet loss, to red cells, with 100% packet loss.

In our next example, shown in Figure 2(b), the visualized matrix for a zone showed sparse failed measurement from all locations. The zone was involved in a denial-of-service attack and the overloaded name servers caused sporadic packet loss. In this type of cases, our tool provides a ready-to-use, close to real-time global overview of the importance of the attack that can help to guide decisions.

The third example involved the “fr.” zone on 15 March 2014. Around 01:30 UTC a high packet loss rate pattern started appearing on DNSMON as depicted in Figure 2(c). Some of the anchors were no longer able to reach the “d.ext.nic.fr” name server because its anycast instance located in Amsterdam was out of order. Our tool played a key role in spotting the issue early on, reducing the reaction time. As visible on the matrix, the instance was quickly restored and the anchors were able to reach again the name server a few hours later the same day. It is conceivable that in other scenarios the problem could be caused by BGP routing problems — where some of the paths reaching the target end up in “routing black holes” — and all the vantage points which end up on these paths share a similar fate. In both scenarios, a comparison of the traceroutes may help to identify where the disruption is located.

TABLE I. MEASUREMENTS USED BY DNSMON, INCLUDING DETAILS OF DNS QUERY OPTIONS AND FREQUENCY.

Type	Protocol	Additional Options	Frequency
CH TXT HOSTNAME.BIND	UDP	No Retries	240s
CH TXT VERSION.BIND	UDP	NSID, IPv4 UDP Payload 1472 bytes, IPv6 UDP Payload 1232 bytes, No Retries	86400s
IN SOA	UDP	NSID, IPv4 UDP Payload 1472 bytes, IPv6 UDP Payload 1232 bytes, No Retries	300s
IN SOA	TCP	No Retries	300s
Traceroute	ICMP		300s

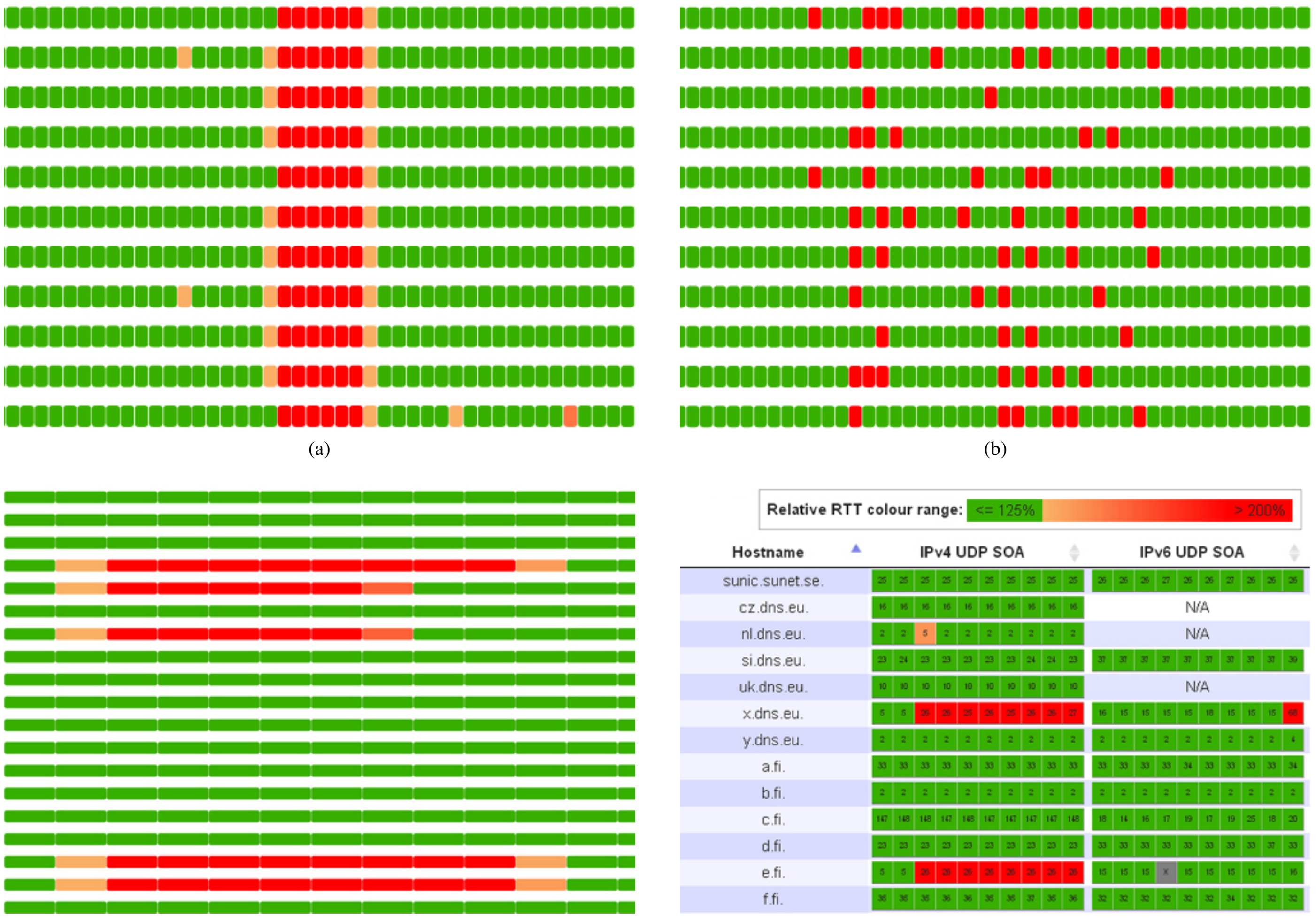
IV. DATA COLLECTION AND AGGREGATION

DNSMON uses the RIPE Atlas measurement network as its data source — in particular, measurement results from RIPE Atlas anchors. The exact set of anchors [17] used in DNSMON is chosen to have as much network and geographical coverage as possible. Currently, anchors are present in all the inhabited continents, especially Europe and the United States, and improving diversity of coverage is a main goal of the project. Each anchor runs the same set of measurements towards the same targets and periodically reports the results back to the RIPE Atlas infrastructure. These results are forwarded and stored in an Apache HBase [18] database that is hosted on an Apache Hadoop [19] cluster. This combination gives us the computational power that we need to process, aggregate and analyze a lot of data, as well as the high availability and good performance required for serving data to interactive clients.

Whenever we want to start monitoring a new zone, we follow these steps: 1) we get the current NS records for this zone; 2) for each record we resolve all available IPv4 and IPv6 addresses; and 3) we start a predefined set of periodic measurements against each address.

The system checks regularly for any changes to the set of the name servers for each zone and, after a manual check, updates the periodic measurements.

RIPE Atlas exposes a range of options per measurement type. The set of measurements and options used by DNSMON is shown in Table I. HOSTNAME.BIND [20] queries reveal which instance of the target service is responding, and are carried out frequently to make it easier to pinpoint routing changes. VERSION.BIND [21] queries reveal the software version of DNS servers, which is unlikely to change often, so they are only carried out once per day. These names are used because of their widespread adoption, but for servers that do not support them, ID.SERVER [20] and VERSION.SERVER would be used instead. The core measurements used by DNSMON are UDP and TCP SOA queries, and are used as the basis for the response times and loss rates in the visualization. The VERSION.BIND and the UDP SOA measurements are configured with the NSID [22] option enabled, which will prompt some servers to include an instance identifier. This may cause the response to be bigger, so we set a UDP response payload size in order to avoid fragmentation issues: 1472 bytes for IPv4 to fit a common MTU of 1500 bytes, minus 28 bytes for the IPv4 and UDP headers; and 1232 bytes for IPv6, to fit an MTU of 1280 bytes, minus 48 bytes for the IPv6 and UDP headers. We perform traceroute measurements mainly



Relative RTT colour range: <= 125% > 200%

Hostname	IPv4 UDP SOA										IPv6 UDP SOA									
sunic.sunet.se.	25	25	25	25	25	25	25	25	25	25	26	26	26	27	26	26	27	26	26	26
cz.dns.eu.	16	16	16	16	16	16	16	16	16	16	N/A									
nl.dns.eu.	2	2	5	2	2	2	2	2	2	2	N/A									
si.dns.eu.	23	24	23	23	23	23	23	24	24	23	37	37	37	37	37	37	37	37	37	39
uk.dns.eu.	10	10	10	10	10	10	10	10	10	10	N/A									
x.dns.eu.	5	5	26	26	25	26	25	26	26	27	16	15	15	15	15	18	15	15	15	68
y.dns.eu.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	4
a.fi.	33	33	33	33	33	33	33	33	33	33	33	33	33	33	34	33	33	33	33	34
b.fi.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
c.fi.	147	148	148	147	148	147	147	147	147	148	18	14	16	17	19	17	19	25	18	20
d.fi.	23	23	23	23	23	23	23	23	23	23	33	33	33	33	33	33	33	33	37	33
e.fi.	5	5	26	26	26	26	26	26	26	26	15	15	15	X	15	15	15	15	15	16
f.fi.	35	35	35	36	36	35	35	37	35	36	32	32	32	32	32	32	34	32	32	32

Figure 2. Examples of problem analysis. In (a), (b) and (c) the main matrix is represented, with x-axis (time) and y-axis (anchors) removed to obscure identities. (a) An outage where all anchors had severe problems reaching the server, causing a thick vertical band. (b) Partial denial-of-service attack, shown as a scattered vertical band. (c) Outage affecting a subset of anchors, with solid horizontal rows. (d) Recent measurements by an anchor.

for investigative purposes as outlined in Section III. At the time of writing this paper, DNSMON monitors 44 zones, by measuring in total 268 servers from 47 anchors worldwide. We have created 2,773 periodic measurements in total, and we store and analyze around 714,000 results daily.

Once collected, the DNS raw results [23] are processed into a streamlined format, containing only the fields necessary for the visualization. The format contains: IDs for the anchor and periodic measurement; the measurement time; an RCODE [24], which may be an error code; and the round-trip time of the query, if completed. Built upon this format are aggregations over 10 minutes, 2 hours and 1 day, i.e., spaced by a factor of twelve. The decision of which levels to use was driven by the requirements of the visualization. There are two types of aggregation (by name server, and by anchor and name server), both of which contain: the number of queries in the period; the number of responses by RCODE, to allow the visualization to distinguish error responses; and the 5th, 50th (median) and 95th percentile of the RTT of all queries that received a reply. In general, aggregations are calculated and stored permanently in HBase tables after a grace period to allow all results to arrive. In order to enable presentation before this point, provisional aggregations are generated on demand and temporarily cached.

V. Implementation and Technical Challenges

In addition to the data collection and aggregation infrastructure, the implementation of DNSMON is split into two other main aspects: a visualization front end and a data API.

The *visualization front end* is a self-contained Web application which can be embedded in any HTML page. It allows the user to view and navigate the interactive matrix. The main interface is presented in Figure 1. It is composed of four main elements: the control bar, the matrix canvas, the data correlation panel, and the time overview bar. We detail their functionality below.

The *control bar* is a toolbar located in the upper part of the interface containing a set of controllers. The first two components from the left are the point of view selector and the colors legend. By changing the point of view, both the matrix and the legend update accordingly. The values for the color thresholds $c1, c2$ specific for the selected point of view are represented in the legend. They can easily be tuned by means of a slider bar appearing by clicking on the legend's labels or on the appropriate button; the change is reflected in real time in the color of the cells. The personalized thresholds are stored locally in the browser to be reused on the next access. The buttons on the right side of the control panel are mainly focused on the time and data navigation, replicating all the gestures accepted by the canvas. The button with the funnel icon allows the user to specify filters on the data, e.g., to exclude: answers containing DNS errors, UDP or TCP, or an IP protocol version. One of the main uses of DNSMON is the continuous monitoring of a zone, for which the visualization must be constantly updated with the latest measurement results without any interaction, e.g., on a wall-mounted monitor where it is essential to be able to see a current overview of a zone a glance. This objective can be achieved by using the *full screen* and *auto update* features accessible from buttons on the control panel. The auto update function keeps the amount of time monitored constant, as well as the data resolution. Newly collected values are introduced as cells on the right side of the matrix, while the old cells are smoothly shifted to the left.

The *matrix canvas* is the main element of the interface. It displays the interactive matrix requested by the user. As described in Section II, the matrix has two axes. The x-axis always represents the selected time interval $\mathcal{T}$, while the y-axis can represent name servers or anchors. All the labels on the y-axis are interactive. Hovering over a label with the mouse provides the user with extra information about the element. A particular name server s can be specified by clicking on a server label, the matrix will automatically switch to the new representation. Alternatively, by clicking on an anchor label, the related anchor page can be opened. In addition to the zoom and shift functionalities, the user can *select a subset of the cells* in order to increase the data resolution on a sub-matrix. We paid particular attention to providing the user with feedback during interaction. While selecting, in addition to the usual selection rectangle, the selected cells are colored in a blue gradient resembling the original tonalities; this gives the user a precise perception of which cells are involved in the selection while keeping the same visual pattern in the background. The transition between two statuses of the matrix is animated.

The *data correlation panel* is hidden by default, appearing when the user clicks on a cell. It provides access to the raw data and to correlated measurement results. In the upper part of the panel, a series of links allow the user to download JSON files from the data API containing the HOSTNAME.BIND, UDP SOA, TCP SOA, traceroute, and VERSION.BIND measurement results collected in the sub-interval represented by the cell. In the lower part of the panel, tabs provide access to: a human-readable DNS response; HOSTNAME.BIND answers collected just before and after the selected cell; and traceroutes collected just before and after the selected cell.

The presentation of the traceroutes is designed to facilitate the kind of analysis shown in Section III by allowing side-by-side comparison, and visually differentiating each line. Traceroute output is enriched with links to get information about IP addresses from RIPEstat [25].

The *time overview bar* is a crucial part of the system, placed at the bottom of the interface. It is a timeline including a resizable range slider that allows the user to select a temporal interval of interest. When the user moves the range slider, the matrix is animated accordingly, downloading or filtering out measurement results. In order to make the interaction easier, a snap to grid is placed along the timeline. With this solution it is easy to precisely select multiple whole days: the selection slider is rounded to the closest tick, accompanied by an animation simulating a magnetic effect. All the components of the system front end are constantly synchronized, providing a consistent perception of time to the user.

The front end is written in JavaScript and HTML. The whole visualization is rendered in the browser using D3.js [26], a Scalable Vector Graphics library. The architecture is based on the Model-View-Controller design pattern [27], while a data layer provides a unified access point to different datasets. The data layer is internally composed of specialized sublayers providing fundamental functionalities, such as format isolation, caching, error handling, and connection management,

The development of DNSMON followed a rigid series of tests. In addition to unit tests, we used virtual machines to test how the web application behaved on different operating system and browser combinations. We had an internal beta stage during which we deployed a *data collection system* gathering usage data, performance scores and errors from the clients. This allowed us to study the users' interaction. The results of our study, together with the introduced improvements, are reported as follows.

1) Usually the user moves in time smoothly; after each interaction only a small portion of the matrix represents new cells. The same applies when the auto update function is active. We designed a client-side cache to minimize redundant calls to the data API, selecting which portion of the matrix to retrieve and combining the results with ones from the cache. Analyzing this solution using the performance logging system, we found that we were able to save up to 88% in network bandwidth.
2) The user is often not decisive during mouse gestures or input selection, trying to correct them incrementally, resulting in useless data calls and redraws. We introduced a layer able to temporarily prevent data calls and to give fake user feedback during the interaction. This gives the user the impression that the gestures are applied in real time, while the redraw with the correct data is done when an anti-flood timer expires.
3) We noticed high memory usage during heavily interactive sections due to the large amount of instantiated cells not yet garbage collected, which prompted us to opt for an ad hoc management of the browser's garbage collector in combination with an Object Pool solution [28].

Due to the large amount of data to display, performance and user experience were two big challenges during the implementation of DNSMON. Different browsers — and different versions of the same browser — are able to handle different numbers of DOM elements, which is why we paid particular attention to always providing the visualization with the right data resolution in order to not overload the browser with an excessive number of data points. With the same test environment described above, we established the number of data points β that the major browsers were able to display in our tool without adversely affecting the user experience and rendering time. In addition to this value, the visualization autonomously computes the maximum number of cells ρ displayable on each row, based on the total number of rows available in the scene, the minimum width possible for a cell, and β. The widget communicates this value to the data API, which provides the best data resolution covering $\mathcal{T}$ with at most ρ data points per row. However, the animations and the real-time feedback are much more expensive than the drawing of the cells. For this reason we introduced a *low-profile mode* able to limit the quality and number of the visual effects while keeping the usability intact. The web application is able to automatically switch to low-profile mode based on the browser version and the actual computational load.

The *data API* is a JSON REST interface. The client provides domain-specific parameters (*zone*, *target*, *TCP/UDP*), which the server translates to periodic measurement IDs and returns aggregated, native or raw data. A major design principle for the API is that it requires minimal knowledge and input from the client. For instance, no a priori knowledge of available resolutions is required for querying, meaning that no client-side changes are required if a new aggregation level is added at a later date, e.g., in order to represent a decade of data. The current levels are listed in the output so that the client can predict a change of resolution when zooming. The client can omit an explicit end time to get the latest data, and possibly specify a length of time instead of an explicit start. The API makes use of a server-side cache. The cached periods are fine grained so as to avoid redundant, overlapping back end queries and cache entries.

VI. Conclusions and Future Work

In this paper, we described a network visualization methodology for monitoring and analyzing the availability and performance of DNS zones worldwide. The incremental information enrichment provided by our graphical metaphor enables the operator to easily move from an overview of the overall quality of a zone or server, to the resolution of single DNS queries, while making it easy to correlate signals from different types of network measurements, e.g., traceroutes. The data correlation occurs in a continuous and persistent way, describing in detail the evolution of a DNS quality issue by providing both real-time and historic measurement results that would otherwise need to be manually collected separately. The flexibility of our visualization, including the ability to switch between different points of view of the same scenario, supports, with a unified interface, the investigation of various issues involving different types of quality degradation.

We demonstrated the effectiveness of this tool at solving real DNS operational problems, by describing strategies and examples in Section III. As future work, we plan to introduce automatic pattern recognition to visually highlight correlations across different targets or vantage points in order to identify a common cause of a hidden problem. Further investigation is warranted into more comprehensive visualization of anycast and load-balanced instances, correlation with routing data, and the extension of the service to any user-defined zone.

References

[1] P. V. Mockapetris, "Domain names - implementation and specification," IETF, RFC 1035, 1987.

[2] J. Postel, "Domain name system structure and delegation," IETF, RFC 1591, 1994.

[3] C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra, "A case for comprehensive dnssec monitoring and analysis tools," Presented at SATIN 2011, UK, http://conferences.npl.co.uk/satin/papers/satin2011-Deccio.pdf, [retrieved: 05, 2015].

[4] P. Ren, J. Kristoff, and B. Gooch, "Visualizing dns traffic," in Proceedings of the 3rd international workshop on Visualization for computer security. ACM, 2006, pp. 23–30.

[5] SRC Cyber, "DNSential Domain Name Service Visualization Tool," http://www.srccyber.com/pdf/C07-050614_DNSentinel.pdf, [retrieved: 05, 2015].

[6] H. Yu, X. Dai, T. Baxley, and J. Xu, "A real-time interactive visualization system for dns amplification attack challenges," in Proceedings of the Seventh IEEE/ACIS International Conference on Computer and Information Science (Icis 2008), ser. ICIS '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 55–60.

[7] G. Di Battista, C. Squarcella, and W. Nagele, "How to visualize the k-root name server," in Graph Drawing. Springer, 2012, pp. 191–202.

[8] RIPE NCC, "DNSMON," http://dnsmon.ripe.net/, [retrieved: 05, 2015].

[9] ——, "Test Traffic Measurement Service," http://www.ripe.net/data-tools/projects/archive/ttm/, [retrieved: 05, 2015].

[10] T. Oetiker, "Round-robin database," http://oss.oetiker.ch/rrdtool/, [retrieved: 05, 2015].

[11] RIPE NCC, "RIPE Atlas," http://atlas.ripe.net/, [retrieved: 05, 2015].

[12] S. Steinarsson, "Downsampling time series for visual representation," Master's thesis, Faculty of Computer Science, University of Iceland, 2013.

[13] B. Shneiderman, "The eyes have it: A task by data type taxonomy for information visualization." Proc. IEEE Symposium on Visual Languages '96, 1996, pp. 336–343.

[14] G. Kistner, "Generating visually distinct colors," http://phrogz.net/css/distinct-colors.html, [retrieved: 05, 2015].

[15] International Organization for Standardization, "ISO 3166-1 alpha-2," https://www.iso.org/obp/ui/, [retrieved: 05, 2015].

[16] M. Hemphill, "A note on adults' color–emotion associations," The Journal of genetic psychology, vol. 157, no. 3, 1996, pp. 275–280.

[17] RIPE NCC. RIPE Atlas anchors used by DNSMON. https://atlas.ripe.net/dnsmon/probes. [retrieved: 05, 2015].

[18] A. HBase, "The apache hadoop project," http://hbase.apache.org/, [retrieved: 05, 2015].

[19] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE, 2010, pp. 1–10.

[20] S. Woolf and D. Conrad, "Requirements for a Mechanism Identifying a Name Server Instance," RFC 4892 (Informational), Internet Engineering Task Force, Jun. 2007.

[21] I. S. Consortium et al., Bind 9 administrator reference manual. Bind, 2005.

[22] R. Austein, "DNS Name Server Identifier (NSID) Option," RFC 5001 (Proposed Standard), Internet Engineering Task Force, Aug. 2007.

[23] RIPE NCC, "RIPE Atlas - raw data structure documentation," https://atlas.ripe.net/docs/data_struct/#v4610_dns, [retrieved: 05, 2015].

[24] Eastlake, et al., "RFC 2929. Domain Name System (DNS) IANA Considerations," http://tools.ietf.org/html/rfc2929, [retrieved: 05, 2015].

[25] RIPE NCC, "RIPEstat," https://stat.ripe.net/, [retrieved: 05, 2015].

[26] M. Bostock, V. Ogievetsky, and J. Heer, "D^3 data-driven documents," Visualization and Computer Graphics, IEEE Transactions on, vol. 17, no. 12, 2011, pp. 2301–2309.

[27] G. E. Krasner, S. T. Pope et al., "A description of the model-view-controller user interface paradigm in the smalltalk-80 system," Journal of object oriented programming, vol. 1, no. 3, 1988, pp. 26–49.

[28] M. Kircher and P. Jain, "Pooling pattern," Proceedings of EuroPlop 2002, 2002, pp. 497–510.

Effective Security Monitoring through System Recognition

Felix von Eye
Leibniz Supercomputing Centre, Munich Network Management Team
Garching n. Munich, Germany
Email:voneye@lrz.de

***Abstract*—The bottleneck of security monitoring is often the huge amount of signatures, which are useless but consume computation power and time. Therefore, the signatures have to be set more accurate for the systems, which should be protected. In this paper, a new approach is presented, which is able to detect more efficient the service and software running on a server. This knowledge helps to select the relevant signatures for the security monitoring, which leads to a more efficient usage of the system's resources.**

***Keywords*–security monitoring; network security; security management; proactive scan; netflows; flow records.**

I. Introduction

Security threats are a challenge for every IT infrastructure. To deal with this threat, there are a lot of different approaches introduced in the past. In general there are three different categories of detection: signature based, anomaly or behavior based, and visualization based [1]. In real world scenarios, only signature based detection systems are widely used, because of the low rate of false positive alarm messages. The anomaly based detection is often used in research or at anti malware companies as this method leads to the detection of new and unknown attacks and malware software. Last but not least, the visualization of system behavior to recognize are at most used in network operation centers with the focus on detection of network anomalies.

In the following, the paper focuses on signature based detection systems. The main drawback of these solutions is that they have to carry all the signatures, such as the ones from the actual threats and all the past signatures. The reason for this is that is is possible that an attacker to use an old vulnerability to penetrate a system, e.g., the system administrator installs an older vulnerable software or the outdated exploit is functional in other constellations as well.

In intrusion detection systems, i.e., Snort [2], there are tens of thousands of rules for different attack and intrusion scenarios [3]. A similar situation is visible in the analysis of virus scanners. There are at the moment more than ten million signatures, which can detect viruses, trojan horses or other malware [4]. These signatures are defined by analyzing software and attack behavior, e.g., the communication of a bot software with its command and control server.

On one side, it is pleasing that the security scanners support a widespread malware detection but on the other hand, the scanners have to scan through this huge amount of signatures to evaluate a threat. By implication, the more signatures are added to the scanners databases the slower it works. This leads to the paradox situation that security officers have to pick only the signatures they hope to be the most critical ones. But how should they know? As the other signatures are not activated they cannot find anything related to the deactivated signatures.

To deal with this problem, this paper proposes a new concept of choosing signatures in security monitoring. This paper focuses without loss of generality at most parts network security monitoring. In other parts of the security monitoring, the knowledge acquisition, e.g., the system recognition, differs, whereas the underlying ideas remain the same. First of all, it is important to know which systems are running and what services they offer. This information is very important for the second step. In this step, the systems are categorized, which enables the possibility to choose only these signatures, which are related to this category, e.g., if there is an apache web server running, only the signatures related to apaches or/and web servers are activated.

The rest of the paper is structured as follows. In the next Section II we take a look at different methods to gain information about a system. These methods are combined in Section III. The results of this service detection is discussed in Section IV. At the end, Section V gives a short conclusion and an outlook on next steps.

II. Methods of system and service detection

In general, there are four different methods to detect systems and services [5]. On one side, there are passive methods and on the other hand, there are active methods. In both categories, there are intrusive methods and also non intrusive ones. In the following, these methods are presented.

A. Active and intrusive asset detection

The active methods are used by administrators, who directly want to know something about a system and therefore scans the system. By using intrusive methods, the administrators take advantage of the fact that every system contains bugs and security holes. These bugs are mostly unique, so it is possible to determine, what is currently running. For example, if you are able to exploit the vulnerability *CVE-2015-0929* [6], than it is clear that the system is a SerVision HVG Video Gateway with firmware before 2.2.26a78. With more knowledge it is possible to examine more precisely the used firmware or software version.

A collection of usable exploits is for example the metasploit framework [7], which includes in total several thousands of different exploits.

The main drawback is that the usage of vulnerabilities is a very high risk for the system, which should be scanned. Very often, vulnerabilities leads to service faults or crashes. On the other hand, for every administrator in charge of security it should be best practice to patch a vulnerable system as soon as possible, so it is not impossible that after a short period of time, this scanning method doesn't get any results.

B. Active and non intrusive asset detection

In the other active method, the non intrusive active method, the administrator scans a system by using port scanners like nmap [8]. These scanners send some predefined network packets to the other system and wait for the response. Because there are some differences in the implementation between different operating systems, it is possible to determine the fingerprint of each system by analyzing the response packets. With this method, regular scans can be performed also to detect changes in the configurations, e.g., via the tool Dr. Portscan [9].

With this method, it is possible to determine, what services are running on a specific system, but as there are in general no differences between the versions in regard of the way of response, it isn't possible to determine the correct version of a service or operating system. Often there are hello messages from the services, in which they identify themselves, but this information can be set by the local administrator without any side effect and is therefore not trustworthy.

But as pointed out by Mäurer in [10], also port scans can be hazardous for servers, particularly if the port scan is done very fast, e.g., with the tool masscan [11] or Zmap [12], which is able to scan a system with more than 10 Gbps.

C. Passive and non intrusive asset detection

In opposite to active methods, the passive methods use only data, which can be measured in the communication path, e.g., at a router or switch. The common way is to use flow records based on the netflow protocol to determine, which host has a connection to other systems [13]. In general, flow records don't contain any information about the services, but with some assumptions it is possible to determine, what kind of service is hosted on a system. For that, it is helpful that the Internet Assigned Numbers Authority (IANA) reserved a huge amount of ports for specific programs, e.g., in Transmission Control Protocol (TCP) port 22 for the Secure Shell (SSH) service. If there is a connection between two hosts with port 22 involved, the probability is very high that there is really a SSH server running on one host.

This method has no effect on the connection or the availability of the service. On the other hand, it is only possible to detect services and systems, which are really communicating.

A very interesting challenge is to determine which operating system is running on one specific system. Therefore, it is needed to know, which update server are normally used in a environment. In general, the update servers of the Microsoft Windows operating system are in the IP range 65.55.0.0/16, so if one finds a connection between a system and these IP addresses and there is more traffic visible than only a port scanning, then it is likely a Windows system. A little bit more complicated is, if you have on Windows side some Windows Server Update Services (WSUS) running or you have Linux systems, which have possibly different update repository servers installed, or you have systems, which are never or only offline updated.

As there are many services, which have their one update servers, it is possible to have a huge list of potential update

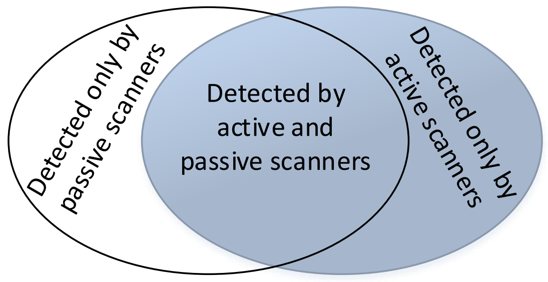

Detected only by passive scanners
Detected by active and passive scanners
Detected only by active scanners

Figure 1. Venn diagram of service detection.

communication, which helps to verify the analysis of the other flow record communication. But it is not possible to determine the version of the used software, as this information isn't transmitted anywhere.

D. Passive and intrusive asset detection

The other possibility to use passive methods is the passive intrusive asset detection, which is done in general via deep packet inspection. In the deep packet inspection, the content of the communication packets are inspected, which enables a very detailed analysis of the communication partners, under the constraint that the communication isn't encrypted with a strong cipher [14]. Similarly, to the port scanners, also at the deep packet inspection it is possible to use some fingerprints to connect the observed communication with a specific service.

This method has no effect on the connection or the availability of the service. On the other hand, it is only possible to detect services and systems, which are really communicating.

As already mentioned, it is difficult to handle encrypted data, but in some cases there are some hints inside the packets, which enable the classification of the traffic without the knowledge about the content of the messages. Furthermore, there are especially in the German and European countries legal constraints in regard of privacy, which don't allow providers to analyze the content of a communication [15].

III. VULNERABILITY MANAGEMENT FOR RULES CREATION

As mentioned before, the most exact method to determine the right type and version of a software is the vulnerability scan, which leads unfortunately sometimes to a system instability. On the other hand, flow record analysis in combination with deep packet inspection enables to detect the fewest amount of software type and in general no version information. But because of routing or firewall restrictions it is most likely that the active scanners are not able to reach any system or service in a network, as passive scanners can only see services, which are used in the detection time frame. Figure 1 shows the overall set of problems in the detection rate.

The biggest challenge in passive detection is that there is no correlation, which enables the administrators to connect the observation of a used connection with the corresponding service. Therefore, we propose the usage of vulnerability scanners to improve the detection on flow record basis. This is achieved with the following steps.

In a first step, all systems are scanned with a vulnerability scanner with a small and controlled rule set. This leads to a list of services and systems, which is reliable. At this point, it is very important that the exploits, which are used in this step, are very well tested to minimize the risk for the connected systems. Furthermore, it is helpful, if the used exploit is not very old. Otherwise, most systems and services are patched by the local administrators in the meantime. But as studies have shown, even for well reported vulnerabilities as the POODLE vulnerability [16], there are four months later even 25 percent of the tested systems still not patched in a bigger university research network [10].

This list of systems and services is now taken to feed the flow record analysis step. From this time on, the flow records of these systems are recorded. As even one service on the host is well known, it is interesting to determine, if the communication from this service can be made visible in the flow records. On the other hand, it is possible to use the knowledge of the deep packet inspection, which allows to determine the used protocol, e.g., with the tool nDPI [17]. This tool is able to detect, for example if a Hypertext Transfer Protocol (HTTP) protocol is used in the communication, even if not the standard port 80 is used. As the deep packet inspection is not able to determine the service, there could be a Linux based Apache web server or a Windows based IIS or other combinations by monitoring a HTTP connection. Nevertheless, the deep packet inspection is very helpful to filter the relevant flow records.

As the relevant flow records are now found, the next step is to compare them in a way, to find some characteristics. So it is noticeable that the response time of an IIS web server is smaller as the same request on an Apache web server. So it is possible to distinguish these two software products by only looking at flow records. As the response time of a web server is in general dependent from the amount of connections per minute, there has to be at least a relationship between the amount of connections and the response times. A fixed value would not be adequate in this case.

Other criteria are the amount of responses to one request, the size of the packets, the delay between packets or also the first bytes of the content of the packet. Depending on the used software, it is possible to detect it only with the usage of deep packet inspection for the used protocol and also flow records for the characteristics.

IV. Using detection results

The results in the detection stage can now be used for the security monitoring. On one side, it is now possible to improve the security incident management [18], by providing more information about a system and the possible break in. On the other hand it is also possible to reduce the total amount of signatures, which are to be used in the security monitoring.

If only the necessary signatures are used, it is possible for a security administrator to use in total more signatures on the same hardware, which leads to total increase of overall security.

In most environments, security monitoring, e.g., network intrusion detection, is done by splitting the traffic on several monitoring points. Mostly the separation is done on port based rules, so all traffic on port 80 is analyzed at one security monitoring point, while the traffic on port 22 is analyzed on another monitoring point. This method is not working very well for huge heterogeneous networks, as not every web server is running on port 80 or 443. The proposed service detection leads to a more exact assignment of systems to monitoring points.

To reach this, all traffic is classified with the above presented method. As the classification allows to determine the exact service, the security monitoring can be adjusted accurately fitting. On the other hand, the monitoring of the flow records and also the deep packet inspection for the protocol detection is not very expensive for monitoring systems.

A very important aspect is, that in regular intervals new vulnerabilities and therefore new classifications are put into account. Due to software updates or changes in protocols and implementations, it is possible that the characteristics are changing.

The main drawback of this approach is the delay until new rules and assignments can be activated. Furthermore, flow records are generally exported regularly after a short amount of time from the switch and analyzed offline, so there is also a measurable delay. A solution could be to transform the approach to a northbound controller application in Software Defined Networks (SDN). There is the possibility that the export of flow records is not needed anymore, as the detection is done by a northbound application at the network controller. This can reduce significantly the delay of the analysis step. But besides from the improvements of using SDN, it is in general unusual that systems and services change very often. Often, there are only small deltas in regard of the different usage of systems, which indicates that the delay is not very important in real world scenarios.

V. Conclusion and Outlook

In this paper, a new approach for the assignment of signature based security monitoring is proposed. This propose was based on the knowledge of vulnerabilities to create rules, which are applicable for flow record detection. This allows administrators to improve the security monitoring of their domain.

In the next steps,we need to define how vulnerabilities, which are used in this approach for crosschecking the results in the rule creation phase, can be processed in an automatic manner. Furthermore, there have to be discovered more characteristics of different services, which allow to use this approach in a wider domain.

Acknowledgments

Parts of this work has been funded by the German Ministry of Education and Research (FKZ: 16BP12309). The authors wish to thank the members of the Munich Network Management (MNM) Team for helpful comments on previous versions of this paper. The MNM-Team, directed by Prof. Dr. Dieter Kranzlmüller and Prof. Dr. Heinz-Gerd Hegering, is a group of researchers at Ludwig Maximilian University of Munich, Technische Universität München, the University of the Federal Armed Forces, and the Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities.

References

[1] Y. Jiang, Z. Jin, A. Abdelkefi, M. Ask, and H. Skrautvol, “Network anomaly detection through traffic measurement,” in Annual Report 2010, S. J. Knapskog, Ed. Trondheim, Norway: Centre for Quantifiable Quality of Service in Communication Systems, Apr. 2011.

[2] M. Roesch. Snort – The Open Source Network Intrusion Detection System. [Online]. Available: http://www.snort.org [retrieved: May, 2015]

[3] G. Münz, N. Weber, and G. Carle, "Signature detection in sampled packets," in Workshop on Monitoring, Attack Detection and Mitigation (MonAM 2007), Toulouse, France, 2007.

[4] F-Secure Corporation. F-Secure DeepGuard^{TM}2.0. Helsinki, Finland. [Online]. Available: https://www.f-secure.com/system/fsgalleries/white-papers/f-secure_deepguard_2.0_whitepaper.pdf [retrieved: May, 2015]

[5] A. Bernhard, "Netzbasierte Erkennung von Systemen und Diensten zur Verbesserung der IT–Sicherheit," Bachelor Thesis, Ludwig Maximilian University of Munich, Munich, Mar. 2014.

[6] NIST – National Vulnerability Database. SerVision HVG Video Gateway web interface contains multiple vulnerabilities. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-0929 [retrieved: Mar., 2015]

[7] Rapid 7. Penetration Testing Software – Metasploit. [Online]. Available: http://www.metasploit.com [retrieved: May, 2015]

[8] G. Lyon, Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Sunnyvale, USA: Insecure.Com, 2008.

[9] W. Hommel, S. Metzger, D. Pöhn, and F. von Eye, "Improving higher education network security by automating scan result evaluation with Dr. Portscan," in ICT Role for Next Generation Universities, ser. EUNIS 2014 – 20th EUNIS Congress, U. Sukovskis, Ed., Umea, Schweden, Jun. 2014, pp. 73–83.

[10] N. Mäurer, "Efficient scans in a research network," Bachelor Thesis, Technische Universität München, Munich, Feb. 2015.

[11] R. D. Graham and P. C. Johnson, "Finite state machine parsing for internet protocols: Faster than you think," in Security and Privacy Workshops (SPW), 2014 IEEE, May 2014, pp. 185–190.

[12] D. Adrian, Z. Durumeric, G. Singh, and J. A. Halderman, "Zippier ZMap: Internet-wide scanning at 10 Gbps," in Proceedings of the 8th USENIX conference on Offensive Technologies. USENIX Association, 2014, pp. 1–8.

[13] M. E. Klepsland, "Passive Asset Detection using NetFlow," Master Thesis, University of Oslo, Department of Informatics, Oslo, Feb. 2012.

[14] T. Böttger, "Detection of Amplification Attacks in Amplifier Networks," Master Thesis, Technische Universität München, Munich, May 2014.

[15] F. von Eye, W. Hommel, and D. Schmitz, "A Secure Logging Framework with Focus on Compliance," in International Journal on Advances in Security, R. Savola, Ed., vol. 7, no. 3 & 4. IARIA, Dec. 2014, pp. 37–49. [Online]. Available: http://www.iariajournals.org/security/sec_v7_n34_2014_paged.pdf

[16] B. Möller, T. Duong, and K. Kotowicz, "This POODLE Bites: Exploiting The SSL 3.0 Fallback." Google, Sep. 2014.

[17] ntop. nDPI – Open and Extensible LGPLv3 Deep Packet Inspection Library. [Online]. Available: https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-0929 [retrieved: May, 2015]

[18] S. Metzger, W. Hommel, and H. Reiser, "Integrated Security Incident Management – Concepts and Real-World Experiences," in 6th International Conference on IT-Security Incident Management and IT Forensics, Stuttgart, May 2011.

Tracking Social Networks Events

Hugo Fonseca, Paulo Salvador and António Nogueira
DETI-University of Aveiro/Instituto de Telecomunicações
Email: {diogolopes, salvador, nogueira}@ua.pt

***Abstract*—Online social media represent a fundamental shift of how information is being produced, transferred and consumed. User generated content in the form of blog posts, comments, and tweets establishes a connection between information producers and consumers. Tracking the pulse of the social media outlets enables companies to gain feedback and insight in how to improve and market products better. For consumers, the abundance of information and opinions from diverse sources helps them make more informed decisions. However, the huge level of online interactions leads to permissive usage behaviours, opening the door for viruses, worms, trojan horses and other threats to install and easily spread, without being noticed. Being able to track users activities, organize the retrieved information in a comprehensive way and analyze it can be very useful for several management, engineering and security tasks. This paper proposes a framework to collect social network events and store them in a relational database for posterior analysis. A graphical user interface was developed to allow flexible access to stored information, according to the type of event, thus facilitating the analysis of users behaviours. From a privacy perspective, the proposed framework is not intrusive because it only gathers the actions timestamps and not their complete contents. By computing statistical models over the obtained data, it is possible to define ”normal or typical” usage profiles and detect possible deviations that can be indicative of a compromised user account.**

Keywords*–*Monitoring Framework; User Behaviour Modelling; Social Network; Compromised User Account.

I. Introduction

Web 2.0 paved the way for the boom of various online social communities, such as Facebook, LinkedIn or Twitter, which successfully facilitate information creation, sharing, and diffusion among users. Nowadays, information sharing and seeking are common user interaction scenarios on the Web. Information can appear in many different forms, like status updates, photos and videos, messages containing comments and/or reactions to certain events. According to the well-known traffic analytics website Alexa [1], the most important Online Social Networks (OSNs) are among the top websites in regard to traffic rankings, being a relevant phenomenon on the Internet.

Understanding the dynamics of these services and being able to track their users activities can be very important for several service design, management, engineering and security tasks. As part of their business model, social media companies make their Application Programming Interfaces (APIs) available to third parties. Besides, some of these companies make their databases on users and usage patterns available through their APIs: using simple software scripts, researchers can access the API to retrieve, store and manipulate digital traces left by the users of a service for further empirical analysis. However, it is not clear to what extent social media APIs can actually offer valid and reliable access points for collecting empirical data. So, APIs should certainly be used together with other methodologies to fully understand the activities and profiles of OSN users and the dynamics of these services.

Cybercriminals can steal information from each social networking profile and its associated posts and, then, tailor their attacks based on the interests and likes of each particular user. This is commonly known as ”social engineering” and makes security threats much more difficult to recognize. A complete analysis of the contents associated to a certain profile allows the identification of compromised accounts, that is, accounts that are under the control of cybercriminals without the knowledge of their licit owners. Illicit contents, containing, for example, links for *phishing* attacks, can be easily disseminated across the social network without raising any suspicion from targeted users.

Social network event collectors, which are able to collect and analyze information posted on a profile, are crucial for the identification of compromised accounts; by collecting the different types of contents posted on each user account, and organizing those events in a timely manner, it becomes easier to identify suspicious events/behaviours.

This paper presents a social network event collector that is able to periodically collect all information posted in a certain profile, as well as all profile related activities (like for example, creation or deletion of new connections, interactions with contents posted by other users, contents the user has shared on his profile). The proposed framework also includes a profile modeling module that can calculate and display several statistical information related to the usage behaviour of any profile. This module interacts with a database containing all collected information for the different user profiles: by computing the average occurrence of certain profile related events and comparing those values with the most recent profile usage statistics, it is possible to detect deviations from a ”normal” behavior in an efficient way. These deviations can be indicative of an illicit profile usage, enabling the detection of possible account hijacks. From a privacy perspective, this framework is not intrusive because it only gathers the actions timestamps and not their complete contents.

The remaining of this paper is organized as follows: Section II presents the most relevant related works; Section III describes the general architecture of the proposed system and discusses its most relevant performance issues; Section IV proposes a methodology to detect usage profile deviations; finally, Section V presents the main conclusions and discusses some possible directions for future work.

II. Related Work

Online social networks have emerged in recent years and became a significant component of the digital life in our society. Several OSNs have millions of users [2][3][4], allowing them to share and find different types of multimedia contents and information. The popularity of these services provides an unique insight into the dynamics of social behaviors.

Mislove et al. [5] proposed a social network growth model based on the analysis of data collected from the Flickr social network. The main obstacle to their approach was the lack of information, which forced the authors to use an API that was made available by Flickr. Motoyama at al. [6] developed a system for searching and matching individuals in order to provide some insight into the dynamics and structures of OSNs. The efficiency of the proposed system was assessed by evaluating the overlap between different OSNs. These results were compared with the ones provided by state-of-the-art matching tools and the authors were able to conclude that their approach provided accurate matching results.

Kazienko et al. and Zanda et al. [7][8] proposed two social network activity recommendation systems. The first implemented a multidimensional social network based on the semantics of the social links between individuals, which enabled the creation of a social recommendation system to present recommendations to the different users. Personal weights are used to allow the system to become personalized and adaptive. The second reference proposed a recommendation system for mobile devices that accesses information from three different sources: mobile, sensor and Facebook. Facebook and mobile data are used to generate a social graph representing the relationships between a user and his friends, or subgroups of friends that can be used to build a recommender for the mobile device. Context data is then used for filtering purposes, allowing the system to present recommendations to the user. These are computed in the mobile device, thus guaranteeing the privacy of sensitive information. However, since the system analyzes the social connections of each user, its complexity depends exponentially on the number of connections.

Beach et al. [9] proposed a system that ties OSNs with mobile devices in an attempt to bind the identity with the corresponding physical location. The goal was also to propose a system on which complex context-aware applications can be built. The basic idea is that users store handles on their mobile devices pointing to their social networking profile that is stored on a remote site. Such profiles, which store information regarding their personalities and preferences as well as the different social network accounts (Facebook, Twitter, LinkedIn and more), are then exchanged between people sharing a common physical place. In this manner, the authors intended to establish a new social interaction paradigm. Several privacy and legal issues were raised by this approach, which also limited its deployment. Besides, the proposed system lacks the ability to create behavior models that could enrich it and turn it in a safer platform.

A framework for reducing the spread of threats between social network users was proposed by Tubi et al. [10]. It comprises a Distributed Network Intrusion Detection System for monitoring the propagation of threats and viruses over the monitored social network. This network was inferred from email addresses obtained from the logs of email servers from Ben-Gurion University. The authors were able to slow down and prevent the propagation of threats by cleaning the traffic from central users of the network. An algorithm for labeling nodes in a social network as honest or illicit was proposed by Danezis et al. [11]. The algorithm, named SybilInfer, uses a probabilistic model of honest OSNs and an inference engine for detecting regions of dishonest nodes. Simulated and real network topologies were used to assess that the algorithm is able to identify compromised nodes. Jin et al. [12] addressed Identity Clone Attacks, which consist of using fake identities for illicit purposes on OSNs. A feasible and effective framework focused on the detection of suspicious identities was proposed. A spectrum based attack detection framework based on the spectral space of underlying network topology was presented by Ying et al. [13]. The work focused on Random Link Attacks, which are filtered by using their spectral coordinate characteristics, which are mainly determined by the coordinates of the victims. The approach improved effectiveness and efficiency when compared to other topology detection approaches. Recently, Cai et al. [14] proposed a statistical model and some associated learning algorithms, named Latent Community model, for the detection of Sybil attacks (where an adversary creates multiple bogus identities to compromise the normal running of the targeted system) on OSNs. This model groups nodes into closely linked communities, which are then linked with the rest of the community. This way, authors could create communities for launching this type of attack, being able to accurately detect them. The results obtained allowed the authors to state that the proposed approach can be the best method for the automatic detection of Sybil attacks. However, the model efficiency should be improved in order to detect low-density attacks.

Perez et al. [15] proposed a dynamic behavioural framework for the identification of suspicious profiles in social networks. The approach is based on three main indicators, balance, energy and anomaly, which are all synthesized from daily user data. Balance refers to the visibility contained within a number of posted messages; energy accounts for the energy that is consumed by a profile for increasing its visibility; the third indicator indicates the anomaly score of an observed activity-visibility pair. The authors argue that suspicious users will have unusual visibility and activity pairs, which is then reflected on the anomaly score. The analysis spans throughout a time period where each indicator is computed and a score is associated to each analysed profile, indicating its level of suspicion. The proposed approach was applied to a set of 2000 Twitter profiles and the obtained results show that suspicious profiles have a more heterogeneous behaviour than normal ones. Moreover, the authors also stated that suspicious profiles present more extreme values of balance, are more likely to spend higher energy than normal profiles and are also likely to have an unusual activity-visibility pairing.

Stringhini et al. [16] analysed spam and spammers in OSNs by creating a large set of ”honey-profiles” and collecting data about spamming. Anomalous users behaviours were identified and a mechanism was proposed to automatically detect spammers. Shen et al. [17] proposed the SOcial network Aided Personalized and effective spam filter (SOAP), where each node connects to its social friends and forms a distributed overlay by using social links, being able to collect information and check spam in an autonomous way. Mahmood [18] identified several privacy leaks of Facebook and Twitter, which allow attackers to collect information for launching targeted attacks such as spam and phishing contents. Solutions for the identified security breaches were also proposed, although their efficiency has not been evaluated. The *SafeGo* tool [19] intends to bring Internet security features to social networking, by protecting users from malware threats that attempt to exploit the trust a user has with his/hers connections. This application uses the BitDefender antimalware and antiphishing engines for scanning Uniform Resource Locators (URLs) through an in-cloud approach based on a blacklist of untrusted URLs. However, unknown threats cannot be detected by this application.

SybilGuard [20] and SybilLimit [21] are two decentralized algorithms for the identification of Sybil attacks in social network topologies. A scalable and efficient solution, named SybilDefender, is able to detect sybil nodes and the underlying community.

III. System Description

The developed system has two main components: the core component, which is coded in Python and Hypertext Preprocessor (PHP), is responsible for estimating/calculating the profiles of the users interactions based on timestamps of events collected from social networks; the web interface component, which is developed in PHP and HyperText Markup Language (HTML) with Cascading Styles Sheets (CSS), allows users to interact and visualize their own profile metrics.

The system architecture is depicted in Figure 1. This distributed architecture dynamically supports several machines, which means that machines can be added or removed at any time. The relational database is used to store metadata corresponding to the different users of each social network, which is extracted by different social network crawling modules. A set of different statistics is then calculated by a Statistics Generator Module and stored on a specific database, the Statistics Database. The Alert Generator Module uses information stored in the Statistics Database to trigger alerts, saving also the data it generates into the same database. Besides these databases, the system also includes a Management Database that is exclusively accessed by the Manager Entity, which is the module responsible for the system coordination.

This framework includes a different relational database for each supported social network. This distributed approach is very important for performance, consistency, troubleshooting and backup issues. Besides, it will also allow system scalability, which is crucial for this kind of continuously evolving tools. Regarding the statistics and alerts database, only one was considered because the amount of data is very small when compared to data retrieved from the OSNs.

The system has a login possibility for each supported online social network. It does not have an autonomous login method, which means that a user is only allowed to login using a social network authentication method. In fact, it does not make sense to have an account on the system for a user that is not a social network user. Since a user can login from several social networks, that is, the system is based on third party authentication, a user has multiple login credentials that have to be associated to only one account. A specific database was created to store information about users identification.

For a particular user, the system starts by fetching information (events timestamps) from all social networks. After this module finishes its execution, the statistics generator module is launched to calculate personal statistics. It is possible to select the social network from which we want to retrieve/check statistics, no matter which social network was used to login. For example, a user can log into the system using Facebook and check his Twitter related statistics, or vice versa. Since the framework supports multiple social networks, the different user profile records should obviously be consistent.

Modules corresponding to a specific user, even if they are from different social networks, run at same time, on the same or on different machines. There are several services

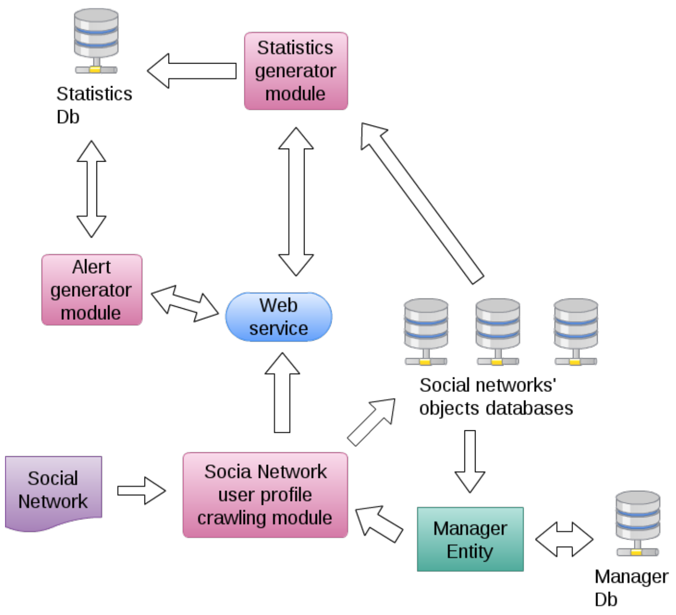

Statistics Db
Statistics generator module
Alert generator module
Web service
Social networks' objects databases
Social Network
Socia Network user profile crawling module
Manager Entity
Manager Db

Figure 1. System Architecture.

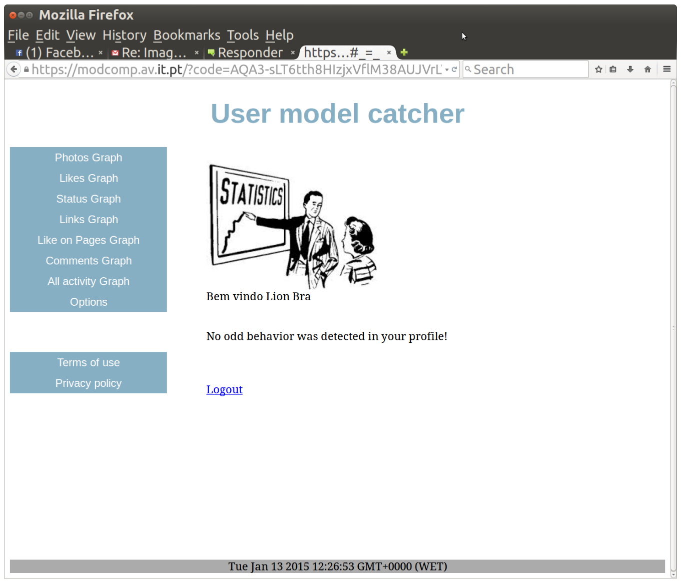
available that allow posting on social networks on behalf of users or collecting and generating users statistics, like for example *http://www.tweetstats.com*, *http://twittercounter.com*, *http://datasift.com* and *http://gnip.com*. Some are paid, others are free, can include more or less functionalities and are mostly used to monitor brands and provide insights into areas like business intelligence, marketing, finance, among others. However, our option was to develop this functionality from scratch in order to avoid being limited on the type data we can get or on the mathematical algorithms that we can apply to the retrieved data.

Figure 2 shows the page that is displayed after a user logs into the system. If there are any alerts, a table showing its details will be presented. Nagivation links are displayed on the left side of the page.

Application users are able to see charts with their total and average amounts of social network activity per day, for a chosen number of days, besides being also able to see the activity of their friends, in some cases. This possibility does not imply any privacy issue because the friend that accepted the application is only allowed to see his current friends on the social network. Indeed, if a friend ceases his relationship, he is no longer able to inspect his ex-friend statistics, unless that ex-friend becomes a friend again on the social network. As illustrative examples of the statistics that can be displayed, Figures 3 and 4 show the number of comments in the last 30 days and the number of all interactions in the last 90 days, respectively.

The developed framework also has the ability to compare the activity of different users, determine which periods correspond to high or low activity and which are the most relevant activity types at each time period (comments, wall status, photos, posted links, likes, etc).

Figures 5 and 6 show some system management functions, namely the ability to add, remove and pause alerts modules, as well as the possibility to manage which users have access to which modules.



Bem vindo Lion Bra

No odd behavior was detected in your profile!

Figure 2. Fist page after user login.

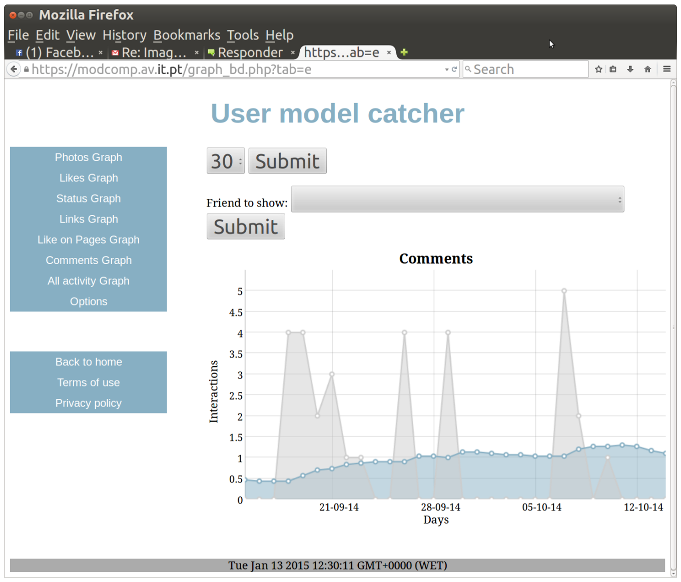


Figure 3. Results - Number of comments in the last 30 days.

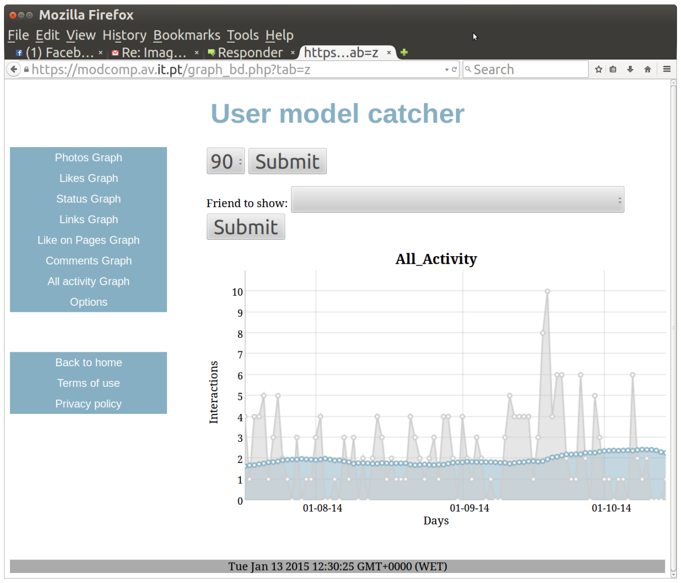
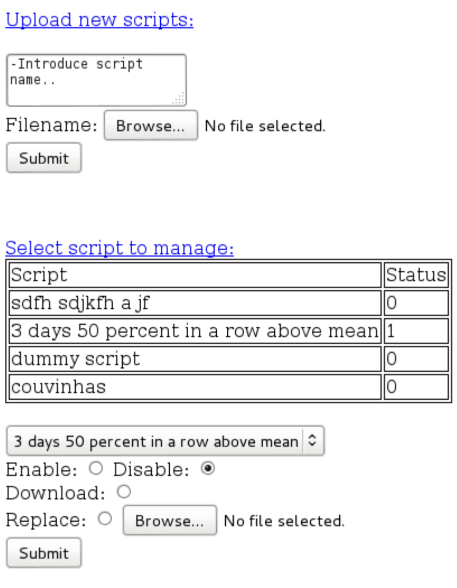


Figure 4. Results - All activities in the last 90 days.


Upload new scripts:

-Introduce script name..

Filename: Browse... No file selected.

Submit

Select script to manage:

Script	Status
sdfh sdjkfh a jf	0
3 days 50 percent in a row above mean	1
dummy script	0
couvinhas	0

3 days 50 percent in a row above mean

Enable: Disable: [✓]

Download:

Replace: Browse... No file selected.

Submit

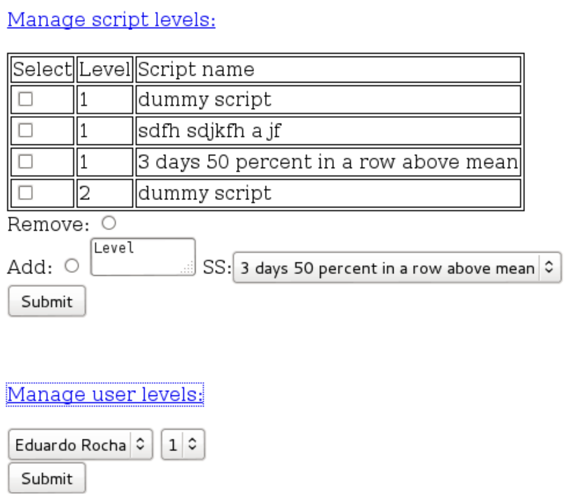
Figure 5. Uploading new management scripts.

The proposed system is based on the timestamps of the social networks events, it does not analyse the contents of user data (photos, text, videos, etc), so user privacy is guaranteed.

Finally, let us briefly discuss system efficiency and performance. The web component is very light and does not require any powerful end device at the user side. At the server side, it is also a light application because it is a PHP webpage with no heavy plugins. Obviously, the performance of the web component will decrease if the number of users accessing the system simultaneously increases significantly.

Core system operations are essentially composed by database accesses and mathematical calculations. Although the core system performance may slightly oscillate depending on the database size, it usually takes from one to two minutes to execute all the operations dedicated to each user.

The retrieval of social network contents is a very slow process: besides the high number of requests and connections that have to be established to retrieve the data, we need to take into account the activity level of social network servers that sometimes may increase download time or even completely prevent data retrieval. To circumvent this drawback, it is necessary to wait for a period of time between connections, thus increasing the time that is needed to complete the process. Another issue that has to be accounted for is the size of the crawled time window: a wider time window usually means more requests, increasing substantially the execution time.


Manage script levels:

Select	Level	Script name
☐	1	dummy script
☐	1	sdfh sdjkfh a jf
☐	1	3 days 50 percent in a row above mean
☐	2	dummy script

Remove: ○

Add: ○ Level SS: 3 days 50 percent in a row above mean

Submit

Manage user levels:

Eduardo Rocha 1

Submit

Figure 6. Managing existing scripts.

IV. Methodology to Detect Profile Usage Deviations

The Alert Generator Module triggers alerts whenever there is a significant difference in user behaviour. Alarms can be immediately sent to the social network profile of each user, if he signs into the application, or sent to his mailbox. For each type of action (comments, likes, etc.), it is necessary to define thresholds for the statistics values.

By defining a set of events $E_t^i = \{e_t^i, t = t_1, ..., t_2\}$ as the set of events of type $i \in \{\mathcal{S}, \mathcal{L}, \mathcal{C}, \mathcal{P}\}$, where each element denotes status updates, likes, comments, photos, respectively, during a time-window of analysis of width $t = t_2 - t_1$, we can compute the average of events of type i within the defined period of analysis as:

$$\overline{E_t^i} = \frac{\sum_{t=t_1}^{t_2} e_t^i}{T_{e_i}} \tag{1}$$

in which T_{e_i} denotes the number of time units (years, months, days,...) considered for analysis within the width of the analysis window defined by $t = t_2 - t_1$. By defining a threshold δ

$$\delta = \alpha.\overline{E_t^i}, \alpha \in \mathbb{R}^+ \tag{2}$$

as a function of the previously defined average, an alarm will be triggered if the average number of events, computed according to (1), within the time period defined by $t' = t_2' - t_1' : t_2' \neq t_2, t_1' \neq t_1$, exceeds the value computed as:

$$E_{t'}^i > \overline{E_t^i} + \delta \tag{3}$$

The threshold can be dynamically redefined in order to decrease/avoid false positives. Threshold δ is defined by the user and, initially, three levels can be considered: aggressive, normal and permissive. In the first level, the tool should be able to detect slight deviations from normal usage profiles; in the normal level, only significant deviations should be detected and in the third level the detection approach is quite permissive, detecting only very important deviation on the usage profiles. Of course, the number of false positives can be quite high in the aggressive mode, so the user can redefine the threshold whenever the detection accuracy falls bellow an acceptable level.

V. Conclusions and Future Work

Online social networks are among the most popular Web sites. Social networking will certainly play an important role in future personal and commercial online interaction, as well as the location and organization of information and knowledge. Examples include browser plug-ins to discover information viewed by friends and social network based, cooperative Web search tools. Social network event collectors can be used to build users behaviour and activity profiles, which are very useful to several management and engineering tasks: evaluate the performance of existing social network platforms, conduct social studies, develop viral marketing strategies, design new content distribution systems, detect anomalous behaviors (such as bots or compromised accounts) and trigger the corresponding alerts, etc. In this paper, we proposed a framework that is able to periodically collect metadata corresponding to all user profile related activities. By retrieving only metadata, no privacy issues are risen by this methodology, because the conducted analysis is only based on timestamps and no activity details are searched and stored in the database. Several online social networks (Facebook, Twitter and Google+) are already supported and the system architecture was designed to be fully compatible with additional social network platforms.

Regarding future developments, we believe that the chat feature of social networks would be interesting to endorse in order to better characterize relationships between users, although this corresponds to a critical privacy issue. We are also planning to incorporate user models based on multi-scale analysis: using concepts that exploit different scales of analysis [22], it is possible to identify the different frequency components that are created by a social network user profile and build mathematical models that can accurately describe the different user interactions.

Acknowledgements

This research is supported by Fundação para a Ciência e Tecnologia through research projects PTDC/EEI-TEL/2016/2012, PEst-OE/EEI/LA0008/2013 and UID/EEA/50008/2013.

References

[1] Alexa traffic statistics for facebook. http://www.alexa.com/siteinfo/facebook.com/. [retrieved: May, 2015]

[2] Welcome to facebook – log in, sign up or learn more. https://www.facebook.com/. [retrieved: May, 2015]

[3] Google+: real life sharing, rethought for the web. https://plus.google.com/up/start/. [retrieved: May, 2015]

[4] Welcome to linkedin. http://www.linkedin.com/. [retrieved: May, 2015]

[5] A. Mislove, H. S. Koppula, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Growth of the flickr social network," in Proceedings of the First Workshop on Online Social Networks, ser. WOSN '08. New York, NY, USA: ACM, 2008, pp. 25–30. [Online]. Available: http://doi.acm.org/10.1145/1397735.1397742

[6] M. Motoyama and G. Varghese, "I seek you: Searching and matching individuals in social networks," in Proceedings of the Eleventh International Workshop on Web Information and Data Management, ser. WIDM '09. New York, NY, USA: ACM, 2009, pp. 67–75. [Online]. Available: http://doi.acm.org/10.1145/1651587.1651604

[7] P. Kazienko, K. Musial, and T. Kajdanowicz, "Multidimensional social network in the social recommender system," Trans. Sys. Man Cyber. Part A, vol. 41, no. 4, Jul. 2011, pp. 746–759. [Online]. Available: http://dx.doi.org/10.1109/TSMCA.2011.2132707

[8] A. Zanda, E. Menasalvas, and S. Eibe, “A social network activity recommender system for ubiquitous devices,” in Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on, Nov 2011, pp. 493–497.

[9] A. A. Beach et al., “Whozthat? evolving an ecosystem for context-aware mobile social networks,” Netwrk. Mag. of Global Internetwkg., vol. 22, no. 4, Jul. 2008, pp. 50–55. [Online]. Available: http://dx.doi.org/10.1109/MNET.2008.4579771

[10] M. Tubi, R. Puzis, and Y. Elovici, “Deployment of dnids in social networks,” in Intelligence and Security Informatics, 2007 IEEE, May 2007, pp. 59–65.

[11] G. Danezis and P. Mittal, “Sybilinfer: Detecting sybil nodes using social networks,” Tech. Rep. MSR-TR-2009-6, January 2009. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=78896

[12] L. Jin, H. Takabi, and J. B. Joshi, “Towards active detection of identity clone attacks on online social networks,” in Proceedings of the First ACM Conference on Data and Application Security and Privacy, ser. CODASPY ’11. New York, NY, USA: ACM, 2011, pp. 27–38. [Online]. Available: http://doi.acm.org/10.1145/1943513.1943520

[13] X. Ying, X. Wu, and D. Barbara, “Spectrum based fraud detection in social networks,” in Data Engineering (ICDE), 2011 IEEE 27th International Conference on, April 2011, pp. 912–923.

[14] Z. Cai and C. Jermaine, “The latent community model for detecting Sybils in social networks,” in Proceedings of the 19th Annual Network & Distributed System Security Symposium, Feb. 2012.

[15] C. Perez, M. Lemercier, and B. Birregah, “A dynamic approach to detecting suspicious profiles on social platforms,” in Communications Workshops (ICC), 2013 IEEE International Conference on, 2013, pp. 174–178.

[16] G. Stringhini, C. Kruegel, and G. Vigna, “Detecting spammers on social networks,” in Proceedings of the 26th Annual Computer Security Applications Conference, ser. ACSAC ’10. New York, NY, USA: ACM, 2010, pp. 1–9.

[17] H. Shen and Z. Li, “Leveraging social networks for effective spam filtering,” IEEE Transactions on Computers, vol. 99, no. PrePrints, 2013, p. 1.

[18] S. Mahmood, “New privacy threats for facebook and twitter users,” in P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2012 Seventh International Conference on, Nov 2012, pp. 164–169.

[19] Bitdefender safego - unfriend your phishy links! http://www.bitdefender.com/solutions/bitdefender-safego.html. [retrieved: May, 2015]

[20] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman, “Sybilguard: Defending against sybil attacks via social networks,” SIGCOMM Comput. Commun. Rev., vol. 36, no. 4, Aug. 2006, pp. 267–278. [Online]. Available: http://doi.acm.org/10.1145/1151659.1159945

[21] H. Yu, P. Gibbons, M. Kaminsky, and F. Xiao, “Sybillimit: A near-optimal social network defense against sybil attacks,” in Security and Privacy, 2008. SP 2008. IEEE Symposium on, May 2008, pp. 3–17.

[22] E. Rocha, P. Salvador, and A. Nogueira, “Detection of illicit network activities based on multivariate gaussian fitting of multi-scale traffic characteristics,” in Communications (ICC), 2011 IEEE International Conference on, June 2011, pp. 1–6.

Multi-scaling Analysis of Social Network Users' Profiles

Paulo Salvador, António Nogueira

Instituto de Telecomunicações, Aveiro, Portugal
DETI, University of Aveiro, Portugal
E-mails: salvador@ua.pt, nogueira@ua.pt

***Abstract*—Online social networks are extremely popular services on the Internet, having a relevant impact on the social, political and economical aspects of our daily lives. Being able to characterize social network users based on their temporal behavior and activity profiles can be useful to several engineering, management and design tasks. This paper characterizes the activity profiles of Facebook users, as well as their time evolution, based on data extracted using the Facebook Graph API and corresponding to social networking activity of 200 users during the years of 2011 and 2012. Using multiscale analysis based on a wavelet decomposition of the dataset, the proposed methodology is able to identify three main periodicity trends on the activity profiles: one of the identified profiles exhibits a visible periodicity around the 24 hours time scale (corresponding to users that have a daily interaction with Facebook), a second profile exhibits relevant components in the complete frequency range (intensive interaction) and a third profile includes important low periodicity components (users that tipically use Facebook in time periods higher than 24 hours). By looking at the temporal evolution of the different activity profiles, it is also possible to understand the dynamics of the profile changes.**

Keywords–Facebook users activity; user profile; scalogram; clustering.

I. Introduction

Some Online Social Networks (OSN), such as Facebook, Twitter or Google+, are among the most popular services on the Internet, enabling individual users to connect to other participating users, share and find contents, and disseminate information through the network, while collecting the reactions from other users without any constraints, either geographic or temporal. Several OSN sites provide social links, like for example networks of professionals and contacts (e.g., LinkedIn, Facebook, MySpace) and networks for sharing contents (e.g., Flickr, YouTube). People access OSNs using both traditional personal computers and new emerging mobile devices. So, these services are really having a deep impact on the social, political and economical aspects of our daily lives.

The success of a social network depends on the behaviour of its users. Characterizing users based on their temporal behavior and activity profiles can be useful in many ways: studies of user behavior will allow to evaluate the performance of existing systems, leading to better site designs [1][2] and advertisement placement policies [3]; social studies can exploit models of user behavior and interaction, while viral marketers can use those models to spread their promotions quickly and widely [4][5]; understanding the workload of OSNs can be very important to design the next-generation Internet infrastructure and content distribution systems [6]; activity profiles can also be used for user classification or detection of anomalous behaviors, such as bots or compromised accounts [7].

Characterizing user activity in major OSNs is challenging because the distribution of most user characteristics is very skewed and shows wide variations, while the majority of participating users have a low (to moderate) level of activity. In this paper, we will characterize the activity profiles of Facebook users, as well as their time evolution, in order to understand how their behaviors change over time. The periodicity trends of the activity profiles are identified using multiscale analysis based on a wavelet decomposition of the time series. In fact, it is known that one of the main purposes of multiscaling analysis is to identify the most important time-scales of (pseudo-periodicity) activity and quantify the constancy of that pseudo-periodicity. The conducted study relies on a medium size dataset, which was extracted using the Facebook Graph API [8], containing social networking activity of 200 users during the years of 2011 and 2012.

From the conducted analysis, we were able to identify three types of activity profiles: an activity profile that exhibits a periodicity around the 24 hours time scale, a second profile that exhibits relevant components in the complete frequency range and a profile that includes important low periodicity components. In other words, the first group of users has an activity periodicity roughly equal to one day; the second group presents significant energy components at the different time scales, which means that these users interact intensively with Facebook, being almost permanently active; the third group uses Facebook in time periods that are generally higher than one day.

By looking at the temporal evolution of the different users behaviors, we could identify a global decrease on the number of users corresponding to the first profile and a steady increase on the number of users corresponding to the other two profiles. This means that the number of users that typically have daily access to Facebook is decreasing and those users tend to behave according to second and third profiles. We also tracked how users changed their profiles in the two years period of our study by calculating the number of users that changed between two specific profile types: we could verify that the first profile was actually the one who lost more users: the majority of those users (aroud 33%) changed their profiles to the second type, although around 14% of them changed their profiles to the third type.

The remaining of the paper is organized as follows: Section II presents some relevant related worksin this area; Section III briefly presents the multiscale analysis methodology that will be used to identify the pseudo-periodicity characteristics of the Facebook users' activity, as well as a summarized description of the k-means clustering algorithm; Section IV presents the social network activity analysis that was conducted, together with its most relevant results; Section V presents the main conclusions and discusses some possible directions for future work.

II. Related Work

The huge amount of data generated by social media users has been used to understand their behavior. By analyzing workloads from three information networks, Guo et al. [9] showed that users' posting behavior exhibited strong daily and weekly patterns. Benevenuto et al. [10] used click-stream data from a social network aggregator to compare user behavior across different OSNs. Papagelis et al. [11] investigated the causality between individual behavior and social influence by observing the information diffusion among users. In order to understand how users' news interests change over time, Liu et al. [12] conducted a large scale analysis of Google News users click logs that were conveniently anonymized. These authors developed a Bayesian framework for predicting users' current news interests from the activities of that particular user and the news trends demonstrated in the activity of all users.

Several works have been studying the properties of user interactions in OSNs. Cha et al. [13] analyzed the spread of favorite-marking of Flickr photos, showing that social links are a primary way users find and share information in social media. Also in 2009, Valafar et al. [14] conducted a measurement study on the Flickr OSN that was able to show that only a small fraction of users in the main component of the friendship graph is responsible for the vast majority of user interactions. Wilson et al. [1] were able to show that interaction activity is significantly skewed towards a small portion of each user's social links by analyzing interaction graphs derived from Facebook user traces. Also using Facebook data, Gilbert and Karahalios [15] were able to demostrate that the "strength of ties" varies a lot, ranging from pairs of users who are best friends to pairs of users who even wished they were not friends. Viswanath et al. [16] studied the evolution of activity between Facebook users, investigating how pairs of users in a social network interact and examining how the varying patterns of interaction affect the overall structure of the activity network.

In a very interesting work, Burke et al. [17] analyzed the activities of a huge number of Facebook users in order to study the roles of user interactions. Visible actions, such as wall posts and comments, and silent actions, such as consumption of friend's content, was conveniently quantified and the authors were able to show that a typical Facebook user communicates with a small subset of their entire friendship network, although it usually maintains relationships with a larger group.

Several efforts have been made to characterize and detect malicious forms of interactions in online social networks. Gao et al. [18] the authors were able to quantify and characterize spam campaigns launched using accounts on online social networks, particularly Facebook. Benevenuto et al. [19] built a test collection of real YouTube users and classified them as spammers, promoters and legitimates, based on social and content attributes. Markines et al. [20] proposed a methodology to detect spam on social bookmarking sites, while Webb et al. [21] placed honeypot accounts on MySpace and study the captured social spammers. Grier at al. [22] presented a characterization of spam on Twitter, finding that this OSN is a highly successful platform for coercing users to visit spam pages. Vasconcelos et al. [23], authors analyzed how Foursquare users exploit the tips, dones and to-dos features to uncover different behavior profiles. This study revealed the existence of very active and influential users that seem engaged in posting tips at a large variety of venues while also receiving a great amount of user feedback on them. Besides, the paper also provided evidence of spamming, showing the existence of users that post tips whose contents are unrelated to the nature or domain of the venue where the tips were left. Wang et al. [7] proposed a machine learning approach to distinguish spam bots from normal ones in Twitter. To facilitate the spam bots detection, three graph-based features, such as the number of friends and the number of followers, are extracted to explore the unique follower and friend relationships among users.

User navigation and usage on OSN websites have also been intensively studied in the last years. Schneider et al. [24] analyzed OSN clickstream data extracted from network traffic, identifying typical user navigation patterns; Jiang et al. [25] used traces from a chinese social network to obtain statistics of profile visits, showing that silent interactions are much more prevalent and frequent than visible events, and that profile popularity are uncorrelated with the frequency of content updates. Joinson et al. [26] identified seven unique reasons for users to use Facebook: social connection, shared identities, content, social investigation, social network surfing, and status updating. Using data from Facebook, Burke et al. [2] studied user motivations for contributing to OSN sites. The authors concluded that newcomers who see their friends contributing to the OSN share more content themselves. Furthermore, those who were initially inclined to contribute, receiving feedback and having a wide audience, were also predictors of increased sharing. Chapman and Lahav [27] analyzed the ethnographical differences in the usage of OSNs. Finally, Caverlee and Webb [28] analyzed over 1.9 million MySpace profiles in order to understand who is using such large OSNs networks and how they are being used.

III. Pseudo-periodicity Analysis

The proposed pseudo-periodicity analysis of the social network users' activity is based on a wavelet decomposition through the Continuous Wavelet Transform (CWT) [29]. Let us consider that $x(t), t \in \{t_0, t_0+\Delta, \ldots, t_0+\Delta i\}, i = 1, 2, \ldots$ quantifies the number of activities performed by a user, within a social network, between $t-\Delta$ and t. Using wavelet decomposition, it is possible to analyze the social interaction process $x(t)$ in both time and frequency domains. The CWT of a process $x(t)$ can be defined as [30]:

$$\Psi_x^\psi(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{+\infty}^{-\infty} x(t)\psi^*(\frac{t-\tau}{s})dt \qquad (1)$$

where $*$ denotes the complex conjugation, $\frac{1}{\sqrt{|s|}}$ is used as an energy preservation factor, $\psi(t)$ is the mother wavelet, while τ and s are the translation and scale parameters, respectively. The first parameter is used for shifting the mother wavelet in time, while the second parameter controls the width of the window analysis and, consequently, the frequency that is being analyzed. By varying these parameters, a multiscale analysis of the entire captured process can be performed, providing a description of the different frequency components present in the decomposed process together with the time-intervals where each one of those components is located. A Wavelet Scalogram can be defined as the normalized energy $\hat{E}_x(\tau, s)$

over all possible translations (set $\mathbf{T}$) in all analyzed scales (set $\mathbf{S}$), and is computed as:

$$\hat{E}_x(\tau, s) = 100 \frac{\left|\Psi_x^\psi(\tau, s)\right|^2}{\sum_{\tau' \in \mathbf{T}} \sum_{s' \in \mathbf{S}} \left|\Psi_x^\psi(\tau', s')\right|^2} \qquad (2)$$

To facilitate the analysis of the scalograms and enable the discovery of the different frequency (periodicity) components, we choose to average the normalized energy over time (set $\mathbf{T}$) obtaining the average energy at timescale s:

$$\bar{e}_x(s) = \frac{1}{|\mathbf{T}|} \sum_{\tau \in \mathbf{T}} \hat{E}_x(\tau, s), \forall s \in \mathbf{S} \qquad (3)$$

where $|.|$ represents cardinality of a set. The existence of a peak in the average energy at a low timescale indicates the existence of a high-frequency (high-periodicity) component in the analyzed time-series, while a peak in the average energy at a high timescale corresponds to the existence of a low-frequency (low-periodicity) component.

In order to classify the distinct users behaviors, the unsupervised learning k-means algorithm was applied to the average normalized energies. The k-means algorithm is one of the simplest unsupervised learning algorithms that is used to solve the clustering problem [31]. The number of clusters, k, is fixed *apriori* and their centers are defined, one for each cluster. These centers should be placed in a cunning way, because different locations causes different results: the better choice is to place them as much as possible far away from each other. In our problem, we verified that three clusters were enough to classify the average normalized energies because an hypothetical forth cluster was simply a replication of one of the other clusters.

The next step of the k-means algorithm is to take each point belonging to a given dataset and associate it to the nearest center. When no point is pending, the first step is completed. At this point, we have to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. A new binding has to be done between the same dataset points and the nearest new center. An iterative approach has been generated and, as a result of this loop, we notice that the k centers change their location step by step until no more changes are done. This algorithm aims at minimizing an objective function, known as squared error function, which is given by:

$$J(\mathbf{C}_1, \ldots, \mathbf{C}_k) = \sum_{j=1}^{k} \sum_{i \in \mathbf{C}_i} \left(\|\bar{e}_i - \mu_j\|\right)^2 \qquad (4)$$

where $\bar{e}_i$ is the s-dimensions data point (average normalized energies) for user i, k is the number of cluster centers, sets $\{\mathbf{C}_1, \ldots, \mathbf{C}_k\}$ contain the data points binded to each one of the k clusters, $\{\mu_1, \ldots, \mu_k\}$ is the set of (s-dimensions) cluster centers, and $\|\bar{e}_i - \mu_j\|$ is the Euclidean distance between $\bar{e}_i$ and μ_j.

IV. Social Network Activity Analysis

As as proof of concept, we used a dataset containing social networking activity (specifically, wall posting) in Facebook of a group of 200 users during 2011 and 2012. The dataset was extracted from a group of Facebook friends of the authors of this paper using the Facebook Graph Application Program Interface (API) [8]. Facebook Graph API is a low level HTTP-based API used to retrieve data from Facebook's Social Graph. Data queried using the Facebook Graph API is returned in JavaScript Object Notation (JSON) format, which can be easily post-processed to extract relevant statistics. The data was divided in four datasets, one per semester: Jan-Jun 2011, Jul-Dec 2011, Jan-Jun 2012 and, Jul-Dec 2012. We considered an interval of analysis of 15 minutes ($\Delta = 15$ minutes).

Note that this type of API usually provides well structured data, but are generally limited in terms of which data, how much data, and how often data can be retrieved. Conditions vary significantly between services: in contrast to Twitter, for example, Facebook is quite restrictive in terms of what data can be accessed, but imposes few limits on the request frequency. Besides, we know that companies retain the right to modify or close their data interfaces, which can lead to substantial problems for researchers.

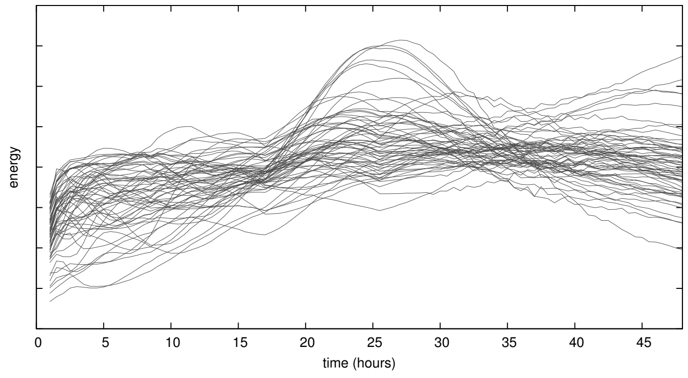


Figure 1. Pseudo-periodicity sample profiles.

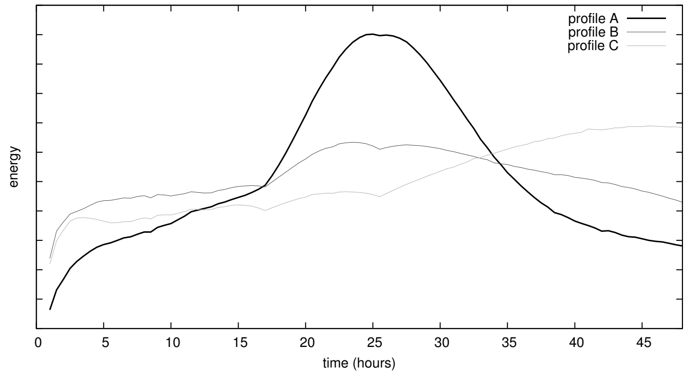


Figure 2. Cluster centroids corresponding to the first semester of 2011.

A. Activity Patterns

Figure 1 shows the average normalized energy coefficients over time for some of the users included in the dataset. Although this plot contains a high number of curves, different profiles can be distinguished: some users have activity profiles that exhibit a noticeable periodicity around the 24 hours time scale (corresponding to the typical bell-shaped curve); for other users, the activity profiles exhibit relevant components in the complete frequency range, while a third group of users include important low periodicity components. The first group of users is the one that presents high energy coefficients around the 24 hours time scales, that is, the periodicity of its activity profile is

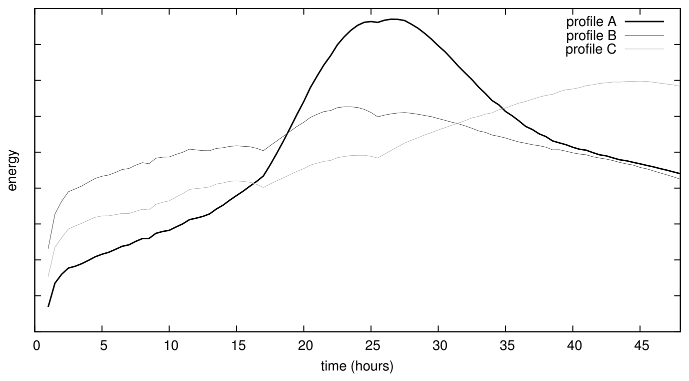


Figure 3. Cluster centroids corresponding to the second semester of 2011.

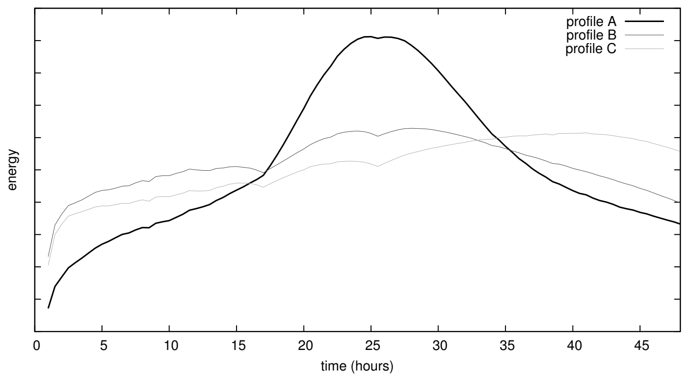


Figure 4. Cluster centroids corresponding to the first semester of 2012.

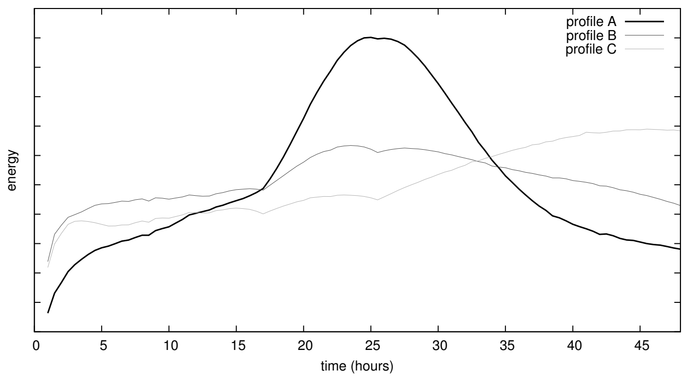


Figure 5. Cluster centroids corresponding to the second semester of 2012.

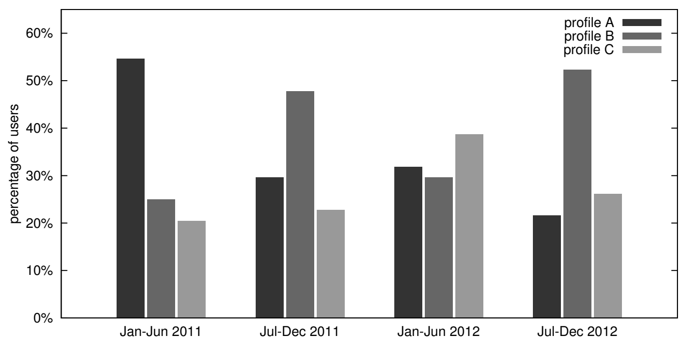


Figure 6. Users' pseudo-periodicity classification over time.

roughly equal to one day. So, these users have typically a daily interaction with Facebook. The second group of users is the one that presents significant energy components at the different time scales, which means that these users interact intensively with facebook, being permanently active. We know that there are several types of Facebook interactions besides wall posting, like messaging, use of different applications, photo upload, chat, among others, but we cannot establish a straighforward relationship between their interaction frequencies. Nevertheless, high activity periods are very likely to include different types of user interactions. Finally, users of the third group use Facebook in a less infrequent way, interacting with the OSN (through wall posting, in this case) in periods that are generally higher than one day. So, these profiles include significant energies for periodicities of several hours.

B. Users Classification

In order to better identify the different activity profiles, the k-means unsupervised clustering algorithm was applied to each one of the four datasets. As previously explained, three clusters were enough to describe the different profiles. In fact, when we used more clusters, some of them tend to overlap.

The different cluster centroids corresponding to the four datasets are presented in Figures 2 to 5. Note that cluster centroids are simply the means of each clustering variable for each cluster. For all datasets, three similar profiles can be clearly identified: profile A exhibits significant energy components around the 24 hours time scale, with lower energy components at the low and high frequency regions of the plot; profile B corresponds to users that include relevant energy components in the complete frequency range; profile C represents users that have significant energy components in the medium to low frequeny range and lower energy components in the high frequency range.

These profiles, clearly revealed by the clustering analysis, are in accordance to our previous analysis. In terms of their wall posting activity, Facebook users can actually be classified into three groups, corresponding to their typical usage behavior.

C. Users Behavior Evolution

Finally, it is important to study the temporal evolution of the different users behavior. Figure 6 shows the size (in terms of the percentage of users) of the different profiles in the four semesters that are included in this study. The main trend that can be identified is a global decrease on the number of users corresponding to profile A and a steady increase (although with some fluctuations) on the number of users corresponding to profiles B and C. This means that the number of users that typically have daily access to Facebook is decreasing; these users tend to behave according to profile B (users that are almost permanently connected) or profile C (users that access to Facebook at longer intervals).

In order to prove this observation, we tried to track how users changed their profiles in the two years period of our study by calculating the number of users that changed between two specific profile types. Figure 7 shows the percentage of users that made a particular change on their profile, for all possible changing combinations. As previously observed, profile A was the one that lost more users: now, we can conclude that the majority of profile A users (aroud 33%) changed their profiles

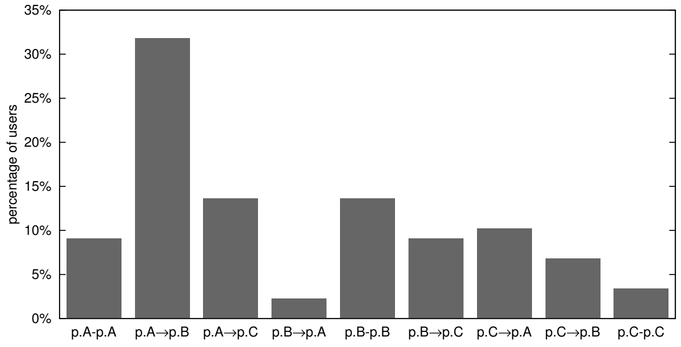


Figure 7. Comparison of users' profiles between Jan-Jun 2011 to Jul-Dec 2012.

to type B, although around 14% of them changed to profile type C. A significantly lower number of users whose profiles were classified as type B changed their profiles: among the ones that changed, the vast majority changed their profiles to type C. Finally, most of the users whose profiles were classified as type C changed their profiles: in this case, a transition to profile type A has a higher probability than a transition to profile type B.

V. Conclusions and Future Work

Some online social networks are among the most popular services on the Internet, having a deep impact on the social, political and economical aspects of our daily lives. Characterizing social network users based on their temporal behavior and activity profiles can be useful to evaluate the performance of existing systems, conduct social studies, develop viral marketing strategies, design new content distribution systems, detect anomalous behaviors, such as bots or compromised accounts, among many other tasks. Based on a medium size dataset, extracted using the Facebook Graph API and containing social networking activity of 200 users during the years of 2011 and 2012, this paper was able to characterize the activity profiles of Facebook users, as well as their time evolution. The periodicity trends of the activity profiles were identified using multiscale analysis based on a wavelet decomposition of the dataset. Three types of activity profiles were identified: an activity profile that exhibits a periodicity around the 24 hours time scale, a second profile that exhibits relevant components in the complete frequency range and a profile that includes important low periodicity components: the first group of users has an activity periodicity roughly equal to one day; users belonging to the second group interact intensively with facebook, while users belonging to the third group use Facebook in time periods that are generally higher than one day. By looking at the temporal evolution of the different users behavior, we were able to identify a global decrease on the number of users corresponding to the first profile and a steady increase on the number of users corresponding to the other two profiles. In fact, the majority of users from the first group (aroud 33%) changed their profiles to the second type, although around 14% of them changed their profiles to the third type.

The identification methodology that was used in this study, which relies on multiscale analysis, will now be applied to the detection of social bots. Social bots, which are automatic or semi-automatic computer programs that mimic humans and/or human behavior in online social networks, can attack users (targets) to pursue a variety of latent goals, such as to harvest private users' data such as email addresses, phone numbers, and other personal data that have monetary value, to spread information or to influence targets. Although many techniques have been proposed to automatically identify spambots in OSNs based on their abnormal behavior, security defenses are not prepared for detecting or stopping a large-scale infiltration caused by social bots.

Acknowledgments

This work was supported by the Fundação para Ciência e Tecnologia (FCT) through PEst-OE/EEI/LA0008/2013, UID/EEA/50008/2013 and PTDC/EEI-TEL/2016/2012.

References

[1] C. Wilson, B. Boe, A. Sala, K. P. Puttaswamy, and B. Y. Zhao, "User interactions in social networks and their implications," in Proceedings of the 4th ACM European Conference on Computer Systems, ser. EuroSys '09. New York, NY, USA: ACM, 2009, pp. 205–218. [Online]. Available: http://doi.acm.org/10.1145/1519065.1519089

[2] M. Burke, C. Marlow, and T. Lento, "Feed me: Motivating newcomer contribution in social network sites," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 945–954. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518847

[3] Y.-M. Li, C.-Y. Lai, and C.-W. Chen, "Discovering influencers for marketing in the blogosphere," Inf. Sci., vol. 181, no. 23, Dec. 2011, pp. 5143–5157. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2011.07.023

[4] T. Rodrigues, F. Benevenuto, M. Cha, K. Gummadi, and V. Almeida, "On word-of-mouth based discovery of the web," in Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 381–396. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068852

[5] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," ACM Trans. Web, vol. 1, no. 1, May 2007. [Online]. Available: http://doi.acm.org/10.1145/1232722.1232727

[6] J. M. Pujol et al., "The little engine(s) that could: Scaling online social networks," SIGCOMM Comput. Commun. Rev., vol. 40, no. 4, Aug. 2010, pp. 375–386. [Online]. Available: http://doi.acm.org/10.1145/1851275.1851227

[7] A. H. Wang, "Detecting spam bots in online social networking sites: A machine learning approach," in Proceedings of the 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy, ser. DBSec'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 335–342. [Online]. Available: http://dl.acm.org/citation.cfm?id=1875947.1875979

[8] "Graph API - Facebook Developers," https://developers.facebook.com/docs/graph-api, 2015, [Online; accessed May-2015].

[9] L. Guo, E. Tan, S. Chen, X. Zhang, and Y. E. Zhao, "Analyzing patterns of user content generation in online social networks," in Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 369–378. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557064

[10] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida, "Characterizing user behavior in online social networks," in Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 49–62. [Online]. Available: http://doi.acm.org/10.1145/1644893.1644900

[11] M. Papagelis, V. Murdock, and R. van Zwol, "Individual behavior and social influence in online social systems," in Proceedings of the 22Nd ACM Conference on Hypertext and Hypermedia, ser. HT '11. New York, NY, USA: ACM, 2011, pp. 241–250. [Online]. Available: http://doi.acm.org/10.1145/1995966.1995998

[12] J. Liu, P. Dolan, and E. R. Pedersen, "Personalized news recommendation based on click behavior," in Proceedings of the 15th International Conference on Intelligent User Interfaces, ser.

IUI '10. New York, NY, USA: ACM, 2010, pp. 31–40. [Online]. Available: http://doi.acm.org/10.1145/1719970.1719976

[13] M. Cha, A. Mislove, and K. P. Gummadi, "A measurement-driven analysis of information propagation in the flickr social network," in Proceedings of the 18th International Conference on World Wide Web, ser. WWW '09. New York, NY, USA: ACM, 2009, pp. 721–730. [Online]. Available: http://doi.acm.org/10.1145/1526709.1526806

[14] M. Valafar, R. Rejaie, and W. Willinger, "Beyond friendship graphs: A study of user interactions in flickr," in Proceedings of the 2Nd ACM Workshop on Online Social Networks, ser. WOSN '09. New York, NY, USA: ACM, 2009, pp. 25–30. [Online]. Available: http://doi.acm.org/10.1145/1592665.1592672

[15] E. Gilbert and K. Karahalios, "Predicting tie strength with social media," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 211–220. [Online]. Available: http://doi.acm.org/10.1145/1518701.1518736

[16] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in Proceedings of the 2Nd ACM Workshop on Online Social Networks, ser. WOSN '09. New York, NY, USA: ACM, 2009, pp. 37–42. [Online]. Available: http://doi.acm.org/10.1145/1592665.1592675

[17] M. Burke, C. Marlow, and T. Lento, "Social network activity and social well-being," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '10. New York, NY, USA: ACM, 2010, pp. 1909–1912. [Online]. Available: http://doi.acm.org/10.1145/1753326.1753613

[18] H. Gao, J. Hu, C. Wilson, Z. Li, Y. Chen, and B. Y. Zhao, "Detecting and characterizing social spam campaigns," in Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 35–47. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879147

[19] F. Benevenuto, T. Rodrigues, V. Almeida, J. Almeida, and M. Gonçalves, "Detecting spammers and content promoters in online video social networks," in Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval, ser. SIGIR '09. New York, NY, USA: ACM, 2009, pp. 620–627. [Online]. Available: http://doi.acm.org/10.1145/1571941.1572047

[20] B. Markines, C. Cattuto, and F. Menczer, "Social spam detection," in Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web, ser. AIRWeb '09. New York, NY, USA: ACM, 2009, pp. 41–48. [Online]. Available: http://doi.acm.org/10.1145/1531914.1531924

[21] S. Webb, J. Caverlee, and C. Pu, "Social honeypots: Making friends with a spammer near you," in Proceedings of Fifth Conference on Email and Anti-Spam, CEAS, Silicon Valley, California, USA, 2008.

[22] C. Grier, K. Thomas, V. Paxson, and M. Zhang, "@spam: The underground on 140 characters or less," in Proceedings of the 17th ACM Conference on Computer and Communications Security, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 27–37. [Online]. Available: http://doi.acm.org/10.1145/1866307.1866311

[23] M. A. Vasconcelos, S. Ricci, J. Almeida, F. Benevenuto, and V. Almeida, "Tips, dones and todos: Uncovering user profiles in foursquare," in Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, ser. WSDM '12. New York, NY, USA: ACM, 2012, pp. 653–662. [Online]. Available: http://doi.acm.org/10.1145/2124295.2124372

[24] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger, "Understanding online social network usage from a network perspective," in Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, ser. IMC '09. New York, NY, USA: ACM, 2009, pp. 35–48. [Online]. Available: http://doi.acm.org/10.1145/1644893.1644899

[25] J. Jiang et al., "Understanding latent interactions in online social networks," in Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 369–382. [Online]. Available: http://doi.acm.org/10.1145/1879141.1879190

[26] A. N. Joinson, "Looking at, looking up or keeping up with people?: Motives and use of facebook," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 1027–1036. [Online]. Available: http://doi.acm.org/10.1145/1357054.1357213

[27] C. N. Chapman and M. Lahav, "International ethnographic observation of social networking sites," in CHI '08 Extended Abstracts on Human Factors in Computing Systems, ser. CHI EA '08. New York, NY, USA: ACM, 2008, pp. 3123–3128. [Online]. Available: http://doi.acm.org/10.1145/1358628.1358818

[28] J. Caverlee and S. Webb, "A large-scale study of myspace: Observations and implications for online social networks," in Proceedings of International Conference on Weblogs and Social Media, Seattle, Washington, USA, 2008.

[29] A. Grossmann and J. Morlet, "Decomposition of hardy functions into square integrable wavelets of constant shape," SIAM Journal on Mathematical Analysis, vol. 15, no. 4, 1984, pp. 723–736.

[30] J. Slavic, I. Simonovski, and M. Boltezar, "Damping identification using a continuous wavelet transform: application to real data," Journal of Sound and Vibration, vol. 262, no. 2, 2003, pp. 291 – 307.

[31] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: analysis and implementation," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 7, Jul 2002, pp. 881–892.