Accepted Manuscript

Title: Exploratory security analytics for anomaly detection

Author: Fabio Pierazzi, Sara Casolari, Michele Colajanni, Mirco Marchetti

PII: S0167-4048(15)00148-0
DOI: http://dx.doi.org/doi: 10.1016/j.cose.2015.10.003
Reference: COSE 949

To appear in: *Computers & Security*

Received date: 16-2-2015
Revised date: 24-7-2015
Accepted date: 2-10-2015

Please cite this article as: Fabio Pierazzi, Sara Casolari, Michele Colajanni, Mirco Marchetti, Exploratory security analytics for anomaly detection, *Computers & Security* (2015), http://dx.doi.org/doi: 10.1016/j.cose.2015.10.003.

Fabio Pierazzi is a Ph.D. student at the International Doctorate School in Information and Communication Technologies (ICT) of the University of Modena and Reggio Emilia, Italy. He received the Master Degree in Computer Engineering from the same university in 2013. His research interests include security analytics and cloud architectures. Home page: http://weblab.ing.unimo.it/people/fpierazzi/
Sara Casolari is a researcher assistant at the Department of Engineering of the University of Modena and Reggio Emilia, Italy. She received her Master Degree and the Ph.D. in Computer Engineering from the same university in 2004 and 2008, respectively. Her research interests include stochastic models and performance evaluation of distributed systems, and modelling algorithms for supporting large systems and Internet-based application. She received a best paper award at the International Conference on Autonomic and Autonomous Systems (ICAS 2007) and at WWW/Internet 2010. Home page: http://weblab.ing.unimo.it/people/sara/
Michele Colajanni is full professor in Computer Engineering at the University of Modena and Reggio Emilia since 2000. He received the Master degree in Computer Science from the University of Pisa, and the Ph.D. degree in Computer Engineering from the University of Roma in 1992. He manages the Interdepartment Research Center on Security and Safety (CRIS), and the Master in "Information Security: Technology and Law". His research interests include security of large scale systems, performance and prediction models, Web and cloud systems. Home page: http://weblab.ing.unimo.it/people/colajanni/
Mirco Marchetti received his Ph.D. in Information and Communication Technologies (ICT) in 2009. He holds a post-doc position at the Interdepartment Center for Research on Security and Safety (CRIS) of the University of Modena and Reggio Emilia. He is interested in intrusion detection, cloud security and in all aspects of information security. Home page: http://weblab.ing.unimo.it/people/marchetti/

Exploratory Security Analytics for Anomaly Detection

Fabio Pierazzi, Sara Casolari, Michele Colajanni and Mirco Marchetti
Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia, Italy
Email address: {fabio.pierazzi, sara.casolari, michele.colajanni, mirco.marchetti}@unimore.it
(Fabio Pierazzi, Sara Casolari, Michele Colajanni and Mirco Marchetti)

Abstract
The huge number of alerts generated by network-based defense systems prevents detailed manual inspections of security events. Existing proposals for automatic alerts analysis work well in relatively stable and homogeneous environments, but in modern networks, that are characterized by extremely complex and dynamic behaviors, understanding which approaches can be effective requires exploratory data analysis and descriptive modeling. We propose a novel framework for automatically investigating temporal trends and patterns of security alerts with the goal of understanding whether and which anomaly detection approaches can be adopted for identifying relevant security events. Several examples referring to a real large network show that, despite the high intrinsic dynamism of the system, the proposed framework is able to extract relevant descriptive statistics that allow to determine the effectiveness of popular anomaly detection approaches on different alerts groups.
Keywords: Security analytics, Network alerts, Temporal characterization, Time series analysis, Anomaly detection.

1. Introduction

The quantity of digital information augments every year and the number of attacks for gaining illegitimate access to these data increases as well. We focus on network defense systems that fire an alert whenever a packet matches a signature related to malware, botnets or any suspicious network activity (e.g., [1, 2, 3]). One of the main issues affecting these systems is the huge number of security alerts generated by sensors that requires a manual inspection by security analysts. In a relatively complex network, it is not uncommon to have from thousands to hundreds of thousands of alerts per day. As a consequence, security analysts tend to focus on a limited portion of the alerts with the risk of missing important events and relations.

Anomaly detection approaches [4, 5] can assist the security analyst in the management of huge volumes of alerts, and are also useful for the identification of relevant events and state changes that may correspond to incidents and infection propagations. However, these techniques work well in relatively stable contexts and for specific datasets, but modern large networked systems are extremely dynamic and heterogeneous [6]. Hence, any automatic and semi-automatic approach for anomaly detection should be combined with preliminary investigations on the observed environment and data. The goal is to understand the main behavioral characteristics and to determine which statistical techniques can be applied legitimately, and which can offer the best results.

This paper focuses on the phases that are often referred to as exploratory data analysis and descriptive modeling [7, 8]. In this context, we propose a novel framework that takes as its input any dataset of security alerts, and is able to discover whether and which anomaly detection algorithms can be effective in the observed environment and data. In particular, the framework investigates temporal trends and patterns in the security alerts, and automatically extracts relevant *descriptive statistics* that are used to understand the applicability of popular anomaly detection approaches (e.g., distribution-based, regression-based [4]). This is achieved by properly separating the alerts in different groups, and by analyzing their data distribution and temporal dependence at different time granularities. The extracted descriptive statistics are also useful for acquiring information about the security status of the observed system, because they allow us to identify the most active classes of alerts, and the most critical subnets and hosts in terms of infection.

To the best of our knowledge, this is the first paper proposing a framework for the automatic investigation of security alerts with the purpose of determining whether and which anomaly detection algorithms can be applied effectively. Our focus on security alerts instead of traffic and netflows information differentiates our proposal from related literature on anomaly detection for security (e.g., [9, 10, 11, 12]). Other works considering temporal analysis and anomaly detection of security alerts either consider outdated datasets [13, 14], or propose novel anomaly detection algorithms that assume specific statistical characteristics of data (e.g., [5, 15, 16]).

We evaluate the proposed framework on real-world alerts referring to a large network environment observed for several months, and we do not make any assumption about the statistical nature of the data. The considered environment is extremely challenging, but approaches that are similar to our agnostic analysis are necessary if you face any modern large network that is dynamic, and affected by several endogenous and exogenous human factors that may intervene on the quantity and type of generated alerts. For example, hosts (dis)connections, activities of network and system administrators on firewall rules, patching and cleaning of hosts. Although all these factors complicate the temporal analysis, our results show that the proposed

framework is able to extract relevant descriptive statistics that allow to understand the effectiveness of popular anomaly detection approaches with respect to different alerts groups.

The remainder of this paper is structured as follows. Section 2 compares the proposal against related work. Section 3 presents the proposed framework and its main phases that are detailed in the successive sections. The first three phases involve preprocessing and partitioning (Section 4), time-based separation of alerts (Section 5), and extraction of descriptive statistics (Section 6). Then, Section 7 describes how from the extracted statistics it is possible to understand which anomaly detection approaches can be applied effectively, and Section 8 presents an evaluation of the anomaly detection choice on real alerts. Section 9 discusses how the proposed framework automatically suggests possible re-aggregations of the alerts for obtaining groups on which anomaly detection could be performed more effectively. Finally, Section 10 outlines conclusions and directions for future research.

2. Related work

Automatic analysis of huge volumes of security alerts represents an increasing research challenge. Existing proposals work well on stable environments, for specific datasets or when they assume a-priori knowledge about possible attack scenarios. These assumptions are becoming less effective in modern networks that are characterized by complex, variable and heterogeneous architectures. Moreover, modern environments present dynamic behaviors at different scales and, in some contexts, many operations on client and mobile hosts are difficult or impossible to be controlled by system administrators. All these reasons induce us and other authors (e.g., [6]) to believe that preliminary studies based on exploratory data analysis and descriptive modeling are necessary as a basis for understanding which (semi-)automatic approaches may be really effective. We propose the first framework for the automatic investigation of temporal trends and patterns in possibly huge volumes of security alerts related to modern networks with the goal of understanding whether and which existing anomaly detection approaches can be applied effectively. Preliminary studies on temporal statistics of the alerts can also help the security analyst in identifying the most critical alerts classes and subnets in terms of network and host infections.

Related analytics efforts are applied in the following three main contexts: attribute-based alerts correlation; anomaly detection in network traffic; anomaly detection in alerts time series.

Most alerts analyses mainly rely on correlations based on similarity between alerts attributes, such as source and destination addresses, or timestamps [17, 18, 19]. Normalization and fusion approaches unify alerts coming from different sources (e.g., IDMEF format [20]). Verification heuristics (e.g., [21, 22]) determine whether an attack related to an alert may be successful or not. Prioritization algorithms [23] associate a level of risk to each alert also by referring to an asset database, where an interesting solution is proposed in [24]. Root cause analysis [25] has the purpose of reducing future alerts by identifying and removing the fundamental causes of alerts. Multi-step attack detection and reconstruction (e.g., [13, 26, 27, 28]) aim at identifying relations between alerts that might be part of the same attack. Most of these correlation approaches work well in stable environments or when a-priori knowledge about possible attacks and alerts statistics is available. The huge number of alerts, the dynamism and complexity of modern networked systems limit the effectiveness of some approaches when applied to real environments.

In this context, anomaly detection approaches [4] can be useful for highlighting the most relevant alerts and state-changes to the security analyst (e.g., possibly corresponding to incidents or infection propagations). The purpose of our framework is to automatically conduct

preliminary temporal investigations of the alerts with the goal of understanding whether and which anomaly detection approaches can be applied effectively on different alerts groups. In [29], the authors propose an original solution for post-correlation of meta-alerts that differs from our paper that focuses on preliminary alerts investigation for anomaly detection purposes. An alerts correlation approach that is more related to our studies since it considers time series analysis for detecting novel attacks without assuming any a-priori knowledge about attack scenarios is presented in [13]. Besides the different goals (multi-step attack detection vs preliminary analyses for anomaly detection), their proposal does not take into account the dynamism intrinsic in modern network environments and their results refer to outdated DARPA datasets [14], while our evaluation is done on real and recent alerts referring to a large network observed for several months.

We observe that we do not consider anomaly detection as a replacement for existing alerts correlation strategies, but rather as a complementary approach to help the security analyst in the management of huge volumes of alerts in modern network environments. Indeed, the preprocessing phase of our framework takes into account the application of existing alerts correlation approaches, for example for unifying alerts coming from different sensors.

Most studies related to anomaly detection [4] that adopt time series analysis [30] do not consider security alerts and mainly focus on traffic and netflows information [9, 10, 11] typically for identifying Distributed Denial of Service attacks [12, 31] and worm propagations [32, 33]. Our work differs from these studies because of the goal and the considered datasets. We are interested in preliminary analyses for understanding the applicability of existing anomaly detection approaches, whereas most literature proposes anomaly detection algorithms where applicability is not really considered or it depends on the nature of the underlying data. Moreover, we consider security alerts instead of network traffic behavior. An advantage of our approach is that each alert is related to a signature corresponding to some malicious activity (e.g., trojan-activity, botnet, scan), while traffic anomalies are more difficult to interpret in highly dynamic contexts. For these reasons, our work is more related to the research of Viinikka et al. that in different papers [5, 15, 165] study anomaly detection of huge volumes of alerts. They are mainly interested in low priority alerts series related to normal system activity (e.g., SNMP and ICMP messages) because some of these signatures generate huge volumes of possibly non-relevant alerts. As these rules are often deactivated or neglected by security analysts, the authors propose different regression-based anomaly detection algorithms for alerts series. Unlike these studies, we do not consider only low priority alerts characterized by strong periodic and trend components, and we do not assume just individual signatures but also different aggregations of alerts. Moreover, our framework identifies which anomaly detection algorithms can be effective on different groups of security alerts; on the other hand, they propose regression-based algorithms that are effective only when applied to series exhibiting strong temporal dependence.

3. Framework overview

The framework proposed in this paper automatically investigates temporal trends and patterns of security alerts to extract relevant *descriptive statistics* that are analyzed to understand effectiveness of popular anomaly detection approaches in the observed environment. Moreover, the proposed framework is also useful for acquiring information about the security status of the observed system for example by identifying the most active classes/signatures of alerts, and the most critical subnets in terms of infection.

Figure 1 reports the main phases of the proposed framework that are identified by white squared boxes:

1. preprocessing and partitioning;
2. time-based separation;
3. extraction of descriptive statistics;
4. analysis of descriptive statistics;
5. identification of possible re-aggregations.

The circles represent *inputs* and *outputs* of the different phases. The initial input is a dataset of security alerts collected by one or more network sensors, and the final output is a set of guidelines for the choice of popular anomaly detection approaches [4] that can be effective on different alerts groups.

The gray boxes in Figure 1 represent *input parameters* that must be chosen by the security analyst. The choice of the input parameters allows to adapt the framework to different contexts and purposes of the analysis. The context determines the amount of alerts available (e.g., one year), the topology of the observed network (e.g., the presence or not of subnets), the number of hosts and the network activity, because higher number of hosts and network activity may correspond to higher number of alerts [5]. The purpose of the analysis depends on the objectives of the security analyst. For example, if he is interested in understanding the applicability of anomaly detection approaches with a responsiveness equal to a hour or a minute, then he must set the time granularity for the analysis accordingly. Moreover, he might be interested in separately monitoring specific subnets or aggregations of internal hosts. Details for the choice of the input parameters are discussed in the following sections.

The first phase of the framework (Figure 1) handles the preprocessing and initial separation of the security alerts into groups. The choice of the groups depends on the topology of the network (e.g., subnets), and on the purposes of the security analyst (e.g., monitoring specific subnets or alerts classes). If the alerts are generated from different network sensors, the security analyst can remove duplicate alerts by combining them through IDMEF [17, 20]. The first phase is discussed in Section 4.

In the second phase, for each alerts group, the framework computes a time series and partitions it on the basis of temporal aggregations (e.g., daytime vs night). This further partitioning is useful to isolate different temporal behaviors (e.g., businessdays vs holidays). If the security analyst has some domain knowledge suggesting possible temporal aggregations for this phase, he can specify them from the beginning. Otherwise, the fifth phase suggests possible re-aggregations by analyzing the descriptive statistics extracted in the third phase. More details on the second phase are discussed in Section 5.

In the third phase, the framework extracts relevant descriptive statistics related to the distribution and temporal dependence of each alerts series. The distribution can be represented through central tendency (e.g., mean, median) and dispersion (e.g., variance, interquartile range, coefficient of variation) of the data. Moreover, this phase also evaluates the stability of alerts distribution statistics over time, as it influences applicability of many anomaly detection approaches [4]. A series exhibits temporal dependence if it has any trend, periodic or seasonal components, or if it may be predicted. Hence, the temporal dependence can be represented through the predictability and/or the periodicity of a series. More details on the third phase are discussed in Section 6.

The fourth phase (Sections 7 and 8) analyzes the extracted descriptive statistics to infer the applicability and effectiveness of popular anomaly detection algorithms. For example,

regression-based anomaly detection (e.g., [16]) could be applied only if the series is predictable [4, 30].

The fifth phase (Section 9) aims to automatically suggest to the security analyst possible re-aggregations of the alerts groups and series that might be more effective for anomaly detection purposes. For example, if the number of alerts is dependent on business hours, then it would be more convenient to extract different descriptive statistics for its distribution (e.g., daytime vs night). This phase might be useful also as a basis to determine time-dependent anomaly detection thresholds.

The upcoming sections present the details of each phase along with examples referring to real-world alerts generated over five months by a network analyzer situated at the edge of a large academic network.

4. Preprocessing and partitioning

The first phase of the framework handles *preprocessing* and *partitioning* of the security alerts. The accepted input may be represented by any type of security dataset, such as raw alerts generated by one or more network sensors, hyper-alerts or meta-alerts resulting from some heuristics for alerts correlation (e.g., [17, 25]) or multi-step attack detection (e.g., [13, 26, 27, 28]). Without loss of generality, as in related literature (e.g., [5]), we focus on raw alerts because they facilitate the identification of temporal patterns that are important for anomaly detection.

The *preprocessing step* of the framework offers to the security analyst the possibility of applying *normalization* and *fusion* algorithms (e.g., [17]) that are required to obtain a unique set of non-duplicated alerts with standardized attributes, such as the IDMEF format [20].

The *partitioning step* separates the security alerts into groups that can be specified by the security analyst through the *initial aggregations* parameter denoted as φ in Figure 1. The choice of these groups depends on the objectives of the security analyst. On the basis of our experience, we suggest to consider at least the following criteria:

a) *alerts origin*, that is, the source address of an alert;

b) *alerts type*, that can refer to common classifications [34] or hyper-alerts [13, 17, 27].

The aggregation criterion based on the *alerts origin* separates the *internal alerts* that are generated from packets issued from internal hosts, and the *external alerts* referring to packets coming from outside the observed network. The motivation for this aggregation should be clear. Internal alerts tend to exhibit temporal behaviors that depend on business hours and users behavior [35], whereas external alerts tend to be more variable and noisy. Finer grained aggregations of the internal alerts can be considered on the basis of the topology of the observed network and the purposes of the analysis. For example, the security analyst can be interested in considering separately groups of hosts subject to different network and firewall policies, such as different subnets/departments or wired/wifi hosts.

The second aggregation criterion based on the *alerts type* is motivated by the observation that different types of attacks or security events tend to exhibit different behaviors. As a consequence, considering all the alerts in one group would prevent any efficacious anomaly detection [5]. For example, in the common case of one class generating a huge number of alerts, the contributions of the other classes would be masked.

The expected output of the first phase of the framework is represented by a set of N alert groups, namely $A_1, A_2, ..., A_i, ..., A_N$ in Figure 1.

Let us now consider an example by referring to an alerts dataset generated by a sensor situated at the edge of a large academic network over five months. For the initial partitioning

with respect to the alerts origin and type, we separate the alerts according to the previously defined criteria:

a) wired, wifi, external;

b) classes of alerts taken by [34].

In particular, we are interested in inspecting wired and wifi alerts separately because most PC clients of internal personnel and all servers have wired connections, while most laptops and all smartphones of internal personnel and guests leverage wireless connections. Moreover, as in most networks, the wifi devices are subject to policies that limit their accesses only to Web and mail applications. For these reasons, we expect different temporal behaviors for the alerts generated from wired and wifi hosts.

The separation of the alerts by type is also relevant to determine the numerosity of each subset. In Table 1 we report the percentage of alerts related to the different classes, with respect to the total number of alerts generated in the observed period. (The classes contributing less than 1% are omitted.) We can observe that more than 80% of the alerts are related to the *trojan-activity* class. This result is credible because in the observed academic environment the network administrators do not have direct control on most of the host devices. The proposed framework can be applied to all alerts groups independently of their numerosity. However automatic analysis is more useful for investigating groups generating huge numbers of alerts, hence in the examples of the next phases we focus on the three most active groups: trojan wired, trojan wifi, trojan external.

5. Time-based separation

The second phase takes as input the alerts groups A_i, $i \in \{1, 2, ..., N\}$ and consists of three main operations that are preliminary for the extraction of descriptive statistics: time series computation, marking series as active/inactive, and time-based partitioning.

For each alerts group A_i, an alerts time series ts_i is computed on the basis of two input parameters:

- *time window w* determining the amount of data to be analyzed (e.g., one month, one year);
- *time granularity g* denoting the temporal unit on which the alerts numerosity should be evaluated (e.g., time series of alerts per day, per hour, or per minute).

These input parameters must be specified by the security analyst, and depend on the context (e.g., amount of available data, level of activity of the observed system) and the objectives of the analysis. For example, if the goal is to find anomalous days or high-level trends in the alerts, then the time granularity *g* could be set equal to a day (so that ts_i contains the number of alerts per day), and the time window *w* equal to six or more months. On the other hand, if the goal is to investigate whether there is a different alerts distribution between daytime and night, the time granularity *g* can be equal to a hour or less, and the time window *w* should be equal to one or more months. In the context of security alerts, excessively fine granularities (e.g., seconds) should be avoided because they cause a lot of noise in the data and prevent the identification of relevant temporal characteristics, such as trends and periodicity [5, 15, 16].

Then, the framework evaluates if each series ts_i is *active* or not in the time window *w*. The purpose of this step is to discard inactive series from further analyses. As a criterion for checking the level of activity of a series, we consider a series *active* if it has generated at least one alert 50% of the time, that is, $median\ (ts_i) > 0$. Other criteria and thresholds for filtering inactive series could be defined depending on the analysis objectives and on the average level of

activity of the observed network. To investigate series marked as inactive, we could consider a coarser time granularity *g*, or a smaller time window *w*.

After the computation of ts_i, and if the series is marked as active, then a further partitioning is performed on the basis of the input parameter *temporal aggregations* δ_i, that is defined as a set of temporal groups (e.g., δ_i={daytime, night}). On the basis of this parameter, the time series ts_i is partitioned in *M* sub-series ts_{ij}, $j \in \{1, 2, ..., M\}$. This further partitioning is useful to isolate possibly different temporal behaviors. If the security analyst has some domain knowledge suggesting possible temporal aggregations for this phase, he can specify δ_i from the beginning. On the other hand, if the security analyst does not have any expectation from the temporal behavior of the alerts, he can specify a fine-grained value of δ_i that is equal for all alerts groups A_i (e.g., that separates the different hourly time-slots of the day). This is motivated by the fact that the fifth phase (Section 9) automatically suggests possible coarser temporal re-aggregations by analyzing the descriptive statistics extracted in the third phase.

The output of this second phase of the framework is a set of *M* sub-series ts_{ij}, along with the complete series ts_i, that is, $M + 1$ time series for each alerts group A_i.

We now present an application of the second phase by referring to the same alerts dataset presented in the previous section, and by focusing on the three most active alerts groups: trojan wired, trojan wifi, and trojan external. We consider a time window *w* equal to five months and a time granularity *g* equal to one hour. This granularity allows us to study the temporal behavior of the alerts in different time-slots (e.g., comparing daytime vs night), and it offers a good trade-off between noise and trends [16]. The time series per hour referring to the trojan wired, trojan wifi and trojan external alerts are reported in Figures 2. The *X*-axis represents the hours, whereas the *Y*-axis reports the number of alerts. In all figures, the *Y*-axis ranges from 0 to 800 alerts per hour. Since all these three series have median values above zero, they are marked as active. From Figures 2 we can observe that the trojan wifi series is the most active. The trojan wired series is characterized by frequent spikes, whereas the trojan external series presents weaker activity.

After computing the series ts_i, we have to specify the input parameter δ_i that defines the temporal aggregations on which the descriptive statistics will be extracted. In the examples of the upcoming phases, we will consider two values of δ_i. On the basis of our domain knowledge, and in order to analyze how the distribution varies depending on the time of the day, and on the day of the week, we first consider δ_i^1 = {businessdays (daytime), businessdays (night), holidays (daytime), holidays (night)}. This aggregation consists of $M = 4$ sub-series ts_{ij} for each alerts group A_i.

As an example for automatic temporal re-aggregations, in Section 9 we consider also a finer grained aggregation δ_i^2 in which ts_i is separated into businessdays and holidays, and then in the 24 time-slots of the day (from 12am to 11pm). This aggregation is useful for studying the similarity of alerts distribution in different time-slots.

In the third phase, the series ts_i and ts_{ij} are used as a basis for the extraction of descriptive statistics for understanding the applicability of anomaly detection approaches on the alerts groups A_i.

6. Extraction of descriptive statistics

The third phase extracts the descriptive statistics useful for understanding the applicability and effectiveness of anomaly detection approaches [4, 36], and for evaluating whether filtering is required [30]. To these purposes, for each alerts group A_i the third phase

takes as input the time series ts_i and the M sub-series ts_{ij} computed in the second phase of the framework. From them, this phase extracts three relevant sets of descriptive statistics, related to *distribution* (Section 6.1), *temporal dependence* (Section 6.2), and *stability* (Section 6.3).

6.1. Distribution

There are two main properties for characterizing a distribution [37]: *central tendency* and *dispersion*. In the considered context that is highly dynamic, we consider the following statistics that are robust to outliers and that can be visually represented through boxplots:

- *median* (*m*) for representing the central tendency of the data;
- *interquartile range* (*iqr*) for representing dispersion around the central tendency.

To represent the impact of outliers on data dispersion, we also consider the *coefficient of variation* $v = \frac{\sigma}{|\mu|}$, where μ and σ are the mean and standard deviation of a distribution/series, respectively. High values of v ($v >> 0$) imply that the series is very dispersed and/or has out-of-scale outliers, whereas low values of v (e.g., $v \approx 0 \div 1$) correspond to a more compact distribution.

We present an example by referring to the same alerts dataset of the previous phases, and to the most active alerts groups: trojan wired, trojan wifi and trojan external. We consider the temporal aggregation $\bar{\delta}_i^1$ defined in Section 5 (that is $\bar{\delta}_i^1$ ={businessdays (daytime), businessdays (night), holidays (daytime), holidays (night)}). Figures 3 report side-by-side boxplots [37] related to $\bar{\delta}_i^1$. The *X*-axis reports the temporal aggregations (e.g., daytime vs night), and the *Y*-axis reports the number of alerts per unit of time (in the example, number of alerts per hour). Each boxplot reports the following statistical properties: lower quartile (q_1), median (m), upper quartile (q_3), interquartile range ($iqr = q_3 - q_1$), lower whisker ($w_l = q_1 - 1.5 \cdot iqr$) and upper whisker ($w_u = q_3 + 1.5 \cdot iqr$). All values above w_u or below w_l are considered outliers.

In Table 2, we report the values of the coefficient of variation v_{ij} related to the different aggregations in $\bar{\delta}_i^1$. This statistic is useful for capturing the variability of the data.

From Figures 3, we can observe that most of the alerts activity is generated by the trojan wifi during businessdays (daytime). On the other hand, the trojan wifi activity is lower during holidays (daytime), and is almost inactive during night. From Table 2, we can observe that the coefficient of variation of the trojan wifi alerts is lower during businessdays (daytime), whereas in the other aggregations it is higher than one, thus indicating that the series is more noisy and/or presents some out-of-scale outliers.

The trojan wired alerts present a similar central tendency (*m*) and dispersion (*iqr*) of the alerts with respect to all four aggregations in Figures 3(a) and 3(d), with a slightly higher activity during businessdays (daytime). However, during businessdays there is a high number of outliers, both during daytime and during night. These outliers are almost one order of magnitude higher than the central tendency, and this is captured by the high values of the coefficient of variation in Table 2.

On the other hand, trojan external alerts are almost equally distributed between daytime and night, with a slightly weaker activity during businessdays (daytime), that is probably related to attacks coming from different time zones. The dispersion of the trojan external alerts is low, and their coefficients of variation are approximately 1.5 in all the temporal aggregations. This suggests that the trojan external series is mainly independent of the detection time, and that it

could be studied as a single aggregation (i.e., without differentiating between businessdays/holidays, daytime/night). This latter conclusion can be automatically achieved by the fifth phase of the framework, as we will discuss in Section 9.

6.2. Temporal dependence

The descriptive statistics related to the temporal dependence are useful to understand the applicability of regression-based anomaly detection [4, 16, 30]. A time series exhibits temporal dependence if it has any trend, periodic or seasonal components in the alerts. The trend represents a general systematic component that does not repeat within the time range captured by the data. For sufficiently long periods, a time series might display periodic or seasonal patterns that repeat in systematic intervals over time.

To extract descriptive statistics about temporal dependence, we consider two popular techniques for the analysis of the time series: *filtering* and *autocorrelation* [30].

Filtering has the objective of reducing the time series noise related to the intrinsic dynamism of any modern network environment that affects alerts generation. This noise might hide trends and temporal patterns that are useful to model the series for anomaly detection purposes. In this phase it is important to consider only simple smoothing filters because more advanced filtering techniques (e.g., [38, 39]) could alter the nature of the data. For this reason, we adopt a *Simple Moving Average* (*SMA*) [30] filter based on a centered window of radius r hours. For the sake of clarity, let us define B_t as an alerts time series, where each element b_t is the number of alerts at time t (e.g., if time granularity g is equal to a day, then b_t represents the number of alerts of day t). The SMA filtering creates a new series $SMA(t)$ where each value of the alerts series b_t is replaced by the mean of b_t and its $2r$ neighbors, as it follows:

$$SMA(t) = \frac{b_{t-r} + b_{t-(r-1)} + \ldots + b_t + \ldots + b_{t+r}}{(2r+1)} \qquad (1)$$

where b_t is an element of the alerts time series at time t, and $(2r + 1)$ is the size of the moving average window. For example, we suggest to consider a radius $r = 1$ for analyzing the effect of a low impact smoothing, and a radius $r = 5$ for a more aggressive filtering.

We also remark that before applying the smoothing filter, it is important to interpolate the values of the most relevant out-of-scale outliers because they could corrupt the autocorrelation analysis and might hide trends or periodicities [30]. For example, outliers above the 99th quantile could be replaced with values corresponding to the upper whisker (w_u) or the upper quartile (q_3) of the distribution.

After filtering, the framework computes the *autocorrelation function* (*ACF*) [30] that is defined as:

$$ACF(\tau) = \frac{E[(B_t - \mu)(B_{t+\tau} - \mu)]}{\sigma^2} \qquad (2)$$

where τ is the lag of the autocorrelation, B_t is the alerts time series per hour, E is the *expected value* operator, μ and σ^2 are the mean and variance of B_t, respectively. A high value and a slow decay of the autocorrelation suggest that future values are related to past values. The opposite is true when the autocorrelation between two values tends to zero. A time series is considered predictable with an adequate accuracy for a prediction window k if its autocorrelation function $|ACF(k)| \geq 0.3$ [30]. Hence, if this last condition is satisfied, regression-based anomaly detection algorithms could be applied effectively.

Unlike the descriptive statistics about the distribution, the temporal dependence statistics are extracted only from the whole series ts_i because the autocorrelation function requires temporal continuity of the data for identifying predictability, trends and periodicity.

In particular, with respect to the temporal dependence we extract the following descriptive statistics:

- the *number of lags* k_i as the number of predictable values;
- the dominant *period* T_i of the time series ts_i (if any).

We observe that there might be more than one period (e.g., 24-hour and 7-day periodicity), but also no periodicity at all (in which case, we consider $T_i = 0$). Moreover, we remark that each of these statistics is extracted with and without the application of smoothing filters to the alerts series ts_i. This means that if we consider three configurations (e.g., no SMA filtering, weak SMA filtering, strong SMA filtering), then there will be three pair of values (k_i, T_i).

We present an application for the extraction of the descriptive statistics about temporal dependence by referring to the same alerts and aggregations considered in the previous sections. In Figures 4 we report the ACFs computed for the trojan wired, trojan wifi and trojan external series ts_i. The *X*-axis represents the lag τ in hours, and the *Y*-axis represents the values of the ACF. The vertical dashed lines denote 24-hour shifts. The horizontal dashed line at 0.3 corresponds to the threshold for determining if a series is predictable or not [30]. Each figure reports the results related to three configurations in order to evaluate whether the conclusions are affected by the choice of the filter: no filtering, SMA filter with radius $r = 1$ and $r = 5$, respectively.

In Figure 4(a), the trojan wired alerts exhibit a weak 24-hour periodicity, that is slightly enhanced by the SMA filtering, but remains below the 0.3 threshold (hence, the period T_i is equal to zero). The filtering slightly improves the prediction lag k_i, especially for $r = 5$, but the series remains weakly correlated. On the other hand, in Figure 4(b) the trojan wifi alerts exhibit a strong 24-hour periodicity, that is evident even without applying the smoothing filters. This means that at each hour the highest probability of finding the same value is every 24 hours ahead. The ACF of the trojan external series (Figure 4(c)) exhibits a trend component that is enhanced by the smoothing filter, reaching values of prediction lag k_i higher than 24 hours for $r = 5$.

6.3. Stability of descriptive statistics

We now propose a method for evaluating the stability of the descriptive statistics related to the alerts distribution, because this information can lead to different conclusions for the applicability of anomaly detection approaches (Section 7).

In order to study the stability of the descriptive statistics of the distribution for each series ts_{ij}, we consider the median m_{ij} and the interquartile range iqr_{ij}. In the second phase of the framework, we have defined *w* as the time window of the dataset to be analyzed (e.g., one year). In this step, we want to verify how the distribution statistics have evolved during the time window *w*. To this purpose, we consider two additional parameters: a *sliding window* of size *s* (e.g., one month) and a *time-shift* Δ (e.g., one week), where $\Delta < s < w$. By changing these parameters, the security analyst can evaluate the stability of descriptive statistics over different time periods. This information is also useful to determine how frequently anomaly detection parameters should be re-estimated. The framework computes the values of inter-quartile range iqr_{ij} and median m_{ij} starting from time interval $t_0 = [0, s]$, then $t_1 = [\Delta, s + \Delta]$, then

$t_2 = [2 \cdot \Delta, s + 2 \cdot \Delta]$, and so on until the time window *w* is covered. The result of this process is a history of the descriptive statistics iqr_{ij} and m_{ij}.

An example is reported in Figures 5 that show the evolution of the descriptive statistics with respect to the alerts dataset considered in the previous sections. The *X*-axis represents the time-shifts *Δ*, and the *Y*-axis represents the values of iqr_{ij} and m_{ij} in terms of number of alerts per hour. In this example, we consider a time window *w* equal to five months, a sliding window *s* equal to one month, and a time-shift *Δ* equal to one week. For example, for X=0 we have the values of m_{ij} and iqr_{ij} computed on the first month. For X=1, we have the values of m_{ij} and iqr_{ij} computed one week ahead, and so on. This allows us to estimate how the descriptive statistics have evolved on a weekly basis.

From Figures 5, we can observe that the trojan wired (daytime) statistics have been unstable in the initial period, but have then stabilized. On the other hand, the trojan wifi alerts always exhibit almost null activity during night, whereas during daytime the activity increases significantly. The trojan external alerts statistics are rather stable for the whole period of observation.

We propose a formal criterion for automatically verifying if the descriptive statistics related to the alerts distribution are stable or not. Let us define *d* as a descriptive statistic (e.g., *iqr*), and d_t as the value of the descriptive statistic *d* at time-shift *t* (e.g., iqr_{ij} at time-shift $5 \cdot \Delta$). In order to evaluate the stability of *d*, we refer to a popular measure of dispersion: the *median absolute deviation* (*MAD*) [40, 41], defined as the median of the absolute deviations from the median. In particular, for each descriptive statistic *d*, we compute a *stability index* β_d that is defined as:

$$\beta_d = \frac{median_t \mid d_t - m(d) \mid}{m(d)} \quad (3)$$

where the numerator represents the MAD, and the denominator $m(d) = median(\{d_0, d_1, ..., d_{T-1}\})$ is a normalization factor required to compare descriptive statistics of different scales. Low values of β_d (near zero) indicate that the descriptive statistic *d* has been stable in the observed period, whereas the opposite is true for higher values of β_d. In particular, the distribution of a time series ts_{ij} is *stable* in central tendency and dispersion if the following relation is satisfied:

$$0 \leq max\{\beta_{m_{ij}}, \beta_{iqr_{ij}}\} \leq \tau \quad (4)$$

where τ is a *stability threshold* that can be adjusted by the security analyst according to the characteristics of dynamism of the observed network environment. In our context, we have heuristically verified that $\tau = 0.2$ is an adequate threshold for automatically discriminating stable and unstable descriptive statistics. In Eq. 4, we consider the maximum between the two stability indexes related to m_{ij} and iqr_{ij}, because large variations of just one descriptive statistics are enough to consider the distribution unstable. The stability indexes related to Figures 5 are reported in Table 3, where values higher than τ are reported in bold. Our framework identifies the distributions related to the wired (daytime) and wifi (daytime) as unstable, whereas the indexes of the other four distributions are lower than the threshold.

7. Analysis of descriptive statistics

The fourth phase analyzes the descriptive statistics extracted from the third phase with the purpose of evaluating the applicability and effectiveness of existing anomaly detection approaches on the observed alerts series. In this paper, we consider the three most popular

families of anomaly detection algorithms [4]: threshold-based (Section 7.1), regression-based (Section 7.2), and distribution-based (Section 7.3). We first introduce the applicability criteria for each of these anomaly detection families, and in Section 7.4 we present a flow-chart for the decision process of the fourth phase, that shows how by analyzing the descriptive statistics the framework is able to suggest the most effective anomaly detection family for each alerts series.

7.1. Threshold-based anomaly detection

Threshold-based algorithms detect anomalies whenever a time series overcomes a certain threshold [4]. If we consider the descriptive statistics related to a series ts_{ij}, a threshold-based approach would be effective for identifying outliers in the data when the distribution of ts_{ij} has a low dispersion. In particular, we consider a ts_{ij} dataset *compact* if its interquartile range is lower than the median, that is, $iqr_{ij} << m_{ij}$ [37]. This criterion can be expressed as the following ratio:

$$\frac{iqr_{ij}}{m_{ij}} < 1 \qquad (5)$$

This ratio is analogous to the definition of coefficient of variation *v*, but it is based on interquartile range instead of standard deviation, and median instead of mean.

If the distribution of ts_{ij} is compact, and the coefficient of variation $v_{ij} \approx 0$, it implies that a threshold based on the first two moments of the distribution (i.e., *mean* and *variance*) would be effective for detecting anomalies. On the other hand, if the distribution is compact and $v_{ij} >> 0$, then the distribution presents some out-of-scale outliers. In this case, threshold-based algorithms require more robust statistics for finding anomalies, such as boxplot statistics or higher order moments [37].

Let us now consider the case in which the distribution is not compact (i.e., $iqr_{ij} >> m_{ij}$). In such a case, threshold-based approaches are unlikely to work well, since the high data dispersion and noise may lead to the detection of an excessively high number of anomalies, possibly corresponding to non-relevant events. In order to understand if a threshold might work well in case of high dispersion, it is necessary to evaluate the *stability* of the descriptive statistics related to the distribution (see Section 6.3). We have two possibilities:

- if iqr_{ij} and m_{ij} are stable over time, then threshold-based approaches have some chances of working well on the observed data, even if data have high variance.
- if iqr_{ij} and/or m_{ij} have frequent changes, then threshold-based approaches will probably lead to the detection of an excessively high number of anomalies possibly corresponding to false positives.

If the descriptive statistics are unstable but follow some trend (e.g., if median and dispersion tend to grow), a possible solution would be to consider *dynamic* thresholds [42] that evolve over time. As a further observation, we consider that if an alerts group A_i exhibits a different distribution between daytime and night, then we suggest to define *multiple* thresholds for detecting anomalies. The number of thresholds should match the number of temporal aggregations defined in δ_i (see Section 5). Section 9 discusses a methodology for unifying similar groups.

We now consider examples of applicability of threshold-based anomaly detection approaches by referring to the alerts dataset considered in the previous sections. In the distributions of trojan wired, trojan wifi and trojan external we have that $iqr_{ij} > m_{ij}$ is always

true. In most cases we have that $\frac{iqr_{ij}}{m_{ij}} \approx 1 \div 2$. This implies that the alerts distribution is not *compact* in any of the classes. Hence, threshold-based approaches have little chances of working well. However, in order to estimate the possibility of applying threshold-based approaches on such alerts series, we evaluate the stability of the descriptive indexes reported in Figures 5 and Table 3. The most stable groups in terms of descriptive statistics are: wired (night), wifi (night), external (daytime), external (night). In these groups, threshold-based algorithms based on robust statistics (e.g., median and inter-quartile range) could be considered for detecting anomalies. It is interesting to observe that in Figure 5(b) the trojan wifi activity has been increasing over time. From this information, the security analyst could consider a *dynamic* threshold that could take into account this growing trend.

7.2. Regression-based anomaly detection

Regression-based approaches assume that a series can be modeled and predicted through some statistical model. Anomalies are detected whenever the value of the prediction *residual* is too high [4, 30].

We recall that for each series ts_i we have extracted the number of predictable lags k_i and the period T_i. The first value of the *ACF* is always equal to one ($ACF(1) = 1$), hence the minimum value of k_i is one, even if there is no temporal dependence in the data [30]. By analyzing the value of k_i, we can understand the applicability of regression-based models for anomaly detection:

- if the ACF decays slowly ($k_i > > 1$), then the series presents a strong trend component, and even *simple regression* algorithms can work well for detecting anomalies;
- if the ACF decays rapidly (k_i is small but higher than 1), then more *complex regression* algorithms should be considered for modeling the temporal dependence appropriately;
- if the ACF decays too rapidly ($k_i \approx 1$), then the temporal dependence of the data is very weak, and regression-based algorithm could be applied for anomaly detection only if the series exhibit a strong periodicity T_i [30].

Examples of *simple regression* algorithms [30] include LR [43] (Linear Regression), OLS (Ordinary Least Squares), MA (Moving Average), WMA (Weighted MA), EWMA [15, 44] (Exponential WMA), and AR model [16] (AutoRegressive model). Examples of *complex regression* algorithms [4] that aim to model more sophisticated relationships between observations are represented by ARMA (AR Moving Average), ARIMA [30] (Integrated ARMA), *robust regression* [45], and others based on Kalman filter [5] or spline interpolation [38].

Let us consider the case in which $k_i \approx 1$ and $T_i = 0$, that is, the series does not exhibit any temporal dependence. This result may be caused by the presence of a strong noise component that is not eliminated by the SMA filter applied automatically in our framework. In this case, more sophisticated and ad-hoc filters may be able to reduce noise and reveal some possible temporal dependence in the data that could be modeled for regression-based anomaly detection. In particular, for each sub-series ts_{ij}, if $iqr_{ij} >> m_{ij}$ and/or $v_{ij} >> 0$, it implies that data is highly dispersed, and strong filtering (possibly including removal of out-of-scale outliers) should be considered for revealing possible temporal dependence. However, the security analyst must be aware that strong filters may alter the nature of the data.

We now consider the applicability of regression-based anomaly detection approaches by referring to Figures 4 that report the autocorrelation results for the trojan wired, trojan wifi and trojan external alerts. The small value of k_i for the trojan wired alerts shows that regression-based approaches would not be effective on this class. On the other hand, the strong 24-hour periodicity of the trojan wifi alerts suggests that regression models can be applied, and that there is a different behavior between daytime and night. The high dispersion of the trojan wifi alerts during daytime (boxplots in Figures 3) suggests that filtering may be useful during daytime hours. Finally, the trojan external series exhibits a trend that corresponds to higher values of k_i, especially when considering the simple smoothing filter with radius $r = 5$, where $k_i \approx 30$ hours.

7.3. Distribution-based anomaly detection

Distribution-based anomaly detection assumes that data can be modeled through some parametric or non-parametric distribution (e.g., Gaussian, Gamma), and that anomalous events occur in low-probability areas of the stochastic model or when the distribution changes significantly [4]. These algorithms can be useful when data exhibit no temporal dependence and regression-based approaches are not applicable.

A series can be modeled through a distribution if its central tendency and dispersion remain stable over time [46]. Hence, if both m_{ij} and iqr_{ij} are stable (see Section 6.3), distribution-based anomaly detection can be applied effectively.

Distribution-based algorithms can be *parametric* or *non-parametric*. Parametric techniques are useful when there is some evidence or knowledge about the underlying distribution of the data. For example, if the median m_{ij} is stable and centered in the interquartile range iqr_{ij}, it is possible that data might be modeled through a *Gaussian* distribution [37], although further analyses would be required, such as χ^2 *goodness-of-fit* test [30, 47, 48]. Other popular parametric distributions are *Gamma* and *Long-tail*. More complex distributions might be even represented or approximated as a *mixture of distributions*, such as *MoG* [4] (Mixture of Gaussians).

Non-parametric techniques are useful when there is no a-priori knowledge about the underlying data distribution [4]. Popular examples are *histogram-based* techniques (e.g., [49]) and *kernel function-based* techniques (e.g., *parzen window estimation* [50]).

As an additional consideration, we observe that if iqr_{ij} is unstable but median m_{ij} is stable, CUSUM-like approaches [51, 52] that adopt the median as descriptive statistics could be effective for anomaly detection.

By referring to Figures 5 and Table 3, we can observe that the trojan external (daytime, night), trojan wifi (night) and trojan wired (night) alerts could be modeled as a distribution, whereas the trojan wifi (daytime) alerts are constantly growing both in mean and in variance, thus complicating the modeling of such alerts series through a distribution. It is interesting to observe that the trojan wired distribution is unstable only in the first period, and then stabilizes. This suggests that distribution-based approaches can be effective on trojan wired after the initial period of instability.

7.4. Flow-chart for the decision process

In Figure 6, we report a flow-chart that merges the applicability criteria together and that suggests the anomaly detection approaches that are most likely to operate on the observed alerts by examining the descriptive statistics collected in the previous phases.

We observe that lower/inner steps of the flow-chart are intended to detect anomalies in more complex and noisy series, that require algorithms with either higher computational

complexities or that are more difficult to configure in terms of parameters. Hence, the flow-chart aims to select the easiest algorithms that are likely to operate for detecting significant anomalies on an observed series.

Let us summarize the decision process by referring to Figure 6. The first step evaluates the *compactness index*: if a series is compact, then threshold-based approaches can be effective for detecting relevant anomalies in the series. In this case, the *coefficient of variation* can be useful to determine if more robust statistics are required if the series presents out-of-scale outliers. On the other hand, if the series is *not compact*, but exhibits temporal dependence, then regression-based approaches can operate well for detecting anomalies. In particular, we recall that if the series exhibits a strong trend component ($k >> 1$), then even simple regression-based approaches could be effective for detecting anomalies on the observed data. If the series has a low prediction lag k but a strong periodicity T, then regression-based approaches could be effective as well, although more sophisticated models are required to represent such periodicity. Finally, if the data does not exhibit any temporal dependence, the stability of the distribution statistics is evaluated: if central tendency and dispersion are stable over time, then distribution-based approaches can be adopted. Otherwise, the framework will try to identify more feasible spatial and temporal re-aggregations of the alerts (details will be discussed in Section 9).

8. Evaluation of the anomaly detection choice

We now evaluate the benefits of the proposed framework in determining the most feasible family of anomaly detection algorithms for different alerts series. We refer to the flow-chart in Figure 6 for determining the most feasible anomaly detection family and to the previous alerts datasets. For each family, we consider a popular anomaly detection algorithm and evaluate which is the most effective for each series.

- *Simple threshold* [4, 37]: an anomaly is detected whenever the alerts series overcomes a threshold determined through the *boxplot rule*;
- *ARIMA* [30] (complex regression-based): an anomaly is detected whenever the prediction error is too high;
- *Histogram-based* [4] (distribution-based): an anomaly is detected through a similarity metric (i.e., *Bhattacharyya coefficient* [53]) between a reference histogram and histograms computed in the other days.

As examples, we apply each of these algorithms to two datasets: trojan alerts in wired and wifi traffic.

8.1. Wired traffic

We consider a series referring to trojan alerts in wired traffic and the flow-chart in Figure 6 to determine which is the most convenient family of anomaly detectors applicable to the considered series. Since it is highly dispersed, threshold-based approaches are expected to be ineffective. As the autocorrelation results show that its prediction lag is low with no periodicity, hence even regression-based algorithms are unlikely to work well. On the other hand, the central tendency and dispersion of this series become stable (see Figures 5), hence distribution-based algorithms are expected to be effective.

We now evaluate whether the conclusions obtained through the flow-chart are correct. To this purpose, in Figures 7 we report the results related to the three considered algorithms: simple threshold, ARIMA, and histogram-based. In these figures, the *lines* represent the same input series and the *markers* on the bottom denote the anomalies signaled by each algorithm.

From Figure 7(a) we can easily observe that the simple threshold algorithm is ineffective because it generates too many detections corresponding to the frequent spikes that characterize the considered series.

The anomalies detected by the *ARIMA* [30] algorithm are reported at the bottom of Figure 7(b). The ARIMA parameters ($p = 3$, $d = 1$, $q = 2$) are determined by applying the *Akaike Information Criterion* [4]. The prediction is performed every 24 hours by considering the filtered series with *SMA* for radius $r = 5$. The bottom part of this figure evidences that the regression-based approach signals many inappropriate anomalies in correspondence of noise. For the sake of clarity, we report in Figure 8 an enlargement of the period going from $X = 978$ to $X = 1584$, where we compare the original alerts series with the predicted series. This figure confirms that a regression-based algorithm is unable to model the trend of the trojan wired series properly. These poor results are consistent with the low autocorrelation results evidenced by the analysis reported in Figure 4(a).

Finally, we consider the histogram-based algorithm as a representative example of distribution-based approaches [4]. We compute a *reference histogram* of alerts occurrences on the whole series, and normalize it between 0 and 1. The result of this process is an histogram that models the most frequent occurrences in the series. Then, we consider the Bhattacharyya Coefficient (*BC*) as a similarity measure for comparing this reference histogram with the normalized histograms computed on the other days of the trojan wired series. In particular, we have $BC \in [0,1]$, where $BC = 1$ implies perfect similarity, whereas $BC = 0$ implies that the compared distributions have nothing in common [53]. If $BC \leq 0.5$, then a day is marked as anomalous. The results of the histogram-based approach are reported in Figure 7(c), where Figure 9 reports the detailed *BC* values computed for the different days of the series. It is interesting to observe that this algorithm is able to find three main groups of anomalies in correspondence of $x_1 \approx 600$, $x_2 \approx 2050$ and $x_3 \approx 2150$. Further inspections on these anomalies reveal that they correspond to infection propagations on new hosts. Hence, as suggested by the flow-chart, the distribution-based anomaly detection has been able to detect the main points in time where hosts got infected, thus confirming that this family of detectors is preferable when the series has the considered features.

8.2. WiFi traffic

We now apply the flow-chart of Figure 6 to the time series corresponding to trojan alerts related to wifi traffic. Given the high dispersion of this series, threshold-based approaches are expected to be ineffective. Despite the small prediction lag, this series presents a strong 24-hour periodicity (Figure 4(b)). Hence, according to the flow-chart, the most feasible family of anomaly detectors should be regression-based. Distribution-based algorithms are expected to be ineffective because if we evaluate also the stability index of this series, we have that it is unstable as its daytime distribution increases over time (see Section 6.3). Let us confirm these conclusions through some experimental analysis.

Figures 10 report the results corresponding to the simple threshold, ARIMA and histogram-based algorithms. The poor results in Figure 10(a) referring to the simple threshold algorithm evidence that it signals too many anomalies. Even by considering different threshold values, there are two intrinsic problems: this series is characterized by a peak in the middle of almost each day, but a threshold signals them as anomalies even if they refer to normal system activities; moreover, the distribution of this alerts group tends to increase over time, hence the number of anomalies detected by the threshold tend to increase over time as well.

Figure 10(c) reports the results related to the histogram-based approach, and Figure 11 shows the values of the Bhattacharyya Coefficient (*BC*). We can observe that the histogram-based approach is ineffective for modeling the distribution of the trojan wifi series, as most *BC* values are situated around the threshold $BC = 0.5$. This is related to the fact that during night most values of the trojan wifi series are close to zero, while during daytime the distribution tends to increase over time. Hence, the distribution-based algorithm is unable to model the distribution of the trojan wifi series appropriately, and cannot detect relevant anomalies effectively. It is also interesting to observe that the number of signaled anomalies tends to increase over time (see Figure 10(c)), because the daytime distribution increases as well.

As anticipated by the flow-chart, the most effective results are achieved by the second class of approaches, here represented by the ARIMA regression-based algorithm, where the parameters ($p = 6$, $d = 1$ and $q = 4$) are determined through the Akaike Information Criterion [4]. In order to better visualize the results shown in Figure 10(b), we also report in Figure 12 a detail of the trojan wifi series, where we compare the filtered series (solid line) with the predicted function (dashed line). We can observe that this approach is able to signal only the most relevant anomalies of the trojan wifi series, with respect to the predicted behavior in a certain period. Hence, in a time frame characterized by higher activity, only the most deviating periods are marked as anomalous.

As a final remark, we can observe that the most effective anomaly detector that better fits the characteristics of the trojan wifi series is represented by the regression-based family, as established by the flow-chart in Figure 6.

9. Possible re-aggregations

This phase of the framework proposes a method for automatically suggesting possible re-aggregations δ_i and φ of the alerts that might provide novel insights when aggregations chosen by the security analyst give little chances of effectiveness for anomaly detection algorithms. As in related literature [54], this phase should be considered as a facilitator in the identification of possible re-aggregations although some manual investigations may still be required. Indeed, completely automatic and unsupervised analyses are almost impossible in modern network domains [6] that are extremely complex and much more dynamic than other security contexts, such as identifying spam in emails [55].

We consider re-aggregations based on *temporal* and *spatial* features.

9.1. Temporal re-aggregation

The temporal re-aggregations may be convenient, for example, if in a group A_i there is a strong difference between daytime and night distribution. Otherwise, if the alerts distribution does not depend on time, then a unique set of statistics for the series ts_i should be considered.

We recall from Section 4 that the parameter δ_i is chosen on the basis of the expectations and domain knowledge of the security analyst (e.g., businessdays vs holidays, daytime vs night), that may not correspond to the actual behavior of the alerts. Hence, this phase could yield a better choice of the parameter δ_i for each alerts group A_i.

The identification of different classes of temporal behavior is carried out through *clustering* [56] on the descriptive statistics related to alerts distribution. In particular, for each alerts group A_i, the framework clusters the descriptive statistics of the M sub-series ts_{ij} related to δ_i. Then, a *cluster separation index* ρ (e.g., [57]) is measured to determine if the clusters are well-separated:

- if ρ is *low*, then δ_i is properly represented by one group because all aggregations have similar distribution.

- if ρ is *high*, then the periodicity T_i is analyzed:
  - if $T_i \neq 0$ (e.g., 24-hour, 7-day), the distribution of the alerts in A_i is dependent on time, and each cluster represents a possible class for the temporal aggregations δ_i;
  - if $T_i = 0$, that is, there is no periodicity, then further analyses are required to determine whether δ_i consists of one or more groups; in this case, some filtering is required because noise may be hiding periodicity in ts_i.

This phase automatically suggests possible re-aggregations, but the input parameter δ_i should be modified manually by the security analyst. We recall from Section 5 that if there is no expectation on possible temporal aggregations, it is better to initialize δ_i by considering fine-grained aggregations (e.g., separate ts_i for each time-slot of the day, from 0 to 23), so that the clustering algorithm can be able to identify groups with similar distribution.

A flow-chart summarizing this phase is presented in Figure 13, where we also observe that if temporal re-aggregation gives unclear results, then spatial re-aggregation can be considered (Section 9.2).

It is important to observe that different clustering algorithms and distance metrics could be adopted for re-aggregation. As the evaluation of an optimal algorithm for identifying temporal aggregations is out of the scope of this paper, in the remainder we show the results referring to the *K-means* algorithm [56]. In Figures 14, we consider as an example the boxplots corresponding to the trojan wired, trojan wifi and trojan external alerts groups, and to the fine-grained aggregation $\overline{\delta_i}^2$. (We recall from Section 5 that $\overline{\delta_i}^2$ considers businessdays vs holidays, and each time-slot of the day separately.) From Figures 14 it is clear that for the trojan wifi alerts there is a strong difference in the alerts distribution between daytime and night, while in the trojan wired and trojan external alerts the distribution is similar in different time-slots of the day (hence, their distribution could be studied as a single group). While this conclusions may be intuitive through a visual representation, the objective of the re-aggregation phase is to automatically detect such a difference. We aim to extract this information automatically through the *K-means* algorithm. Determining the optimal number of clusters *K* for a dataset is a well-known issue in literature [58] and is out of the scope of this paper. Since we are interested in identifying if the alerts follows *at least* two different temporal behaviors (e.g., daytime vs night), or if their distribution is similar independently of the time, for the *K*-means algorithm we consider a parameter $K = 2$. In this way, the *K*-means will separate the hourly time-slots in two clusters: if these two clusters are well separated (i.e., if the separation index ρ is high), then there are *at least* two different temporal behaviors in the alerts distribution (e.g., daytime vs night); on the other hand, if ρ is low, we can conclude that the distribution of the alerts are similar, and could be studied together as a single group.

As an input for the clustering phase, for each of the alerts groups (e.g., wifi businessdays) we consider a two-dimensional feature vector where each feature consists of the pairs (m_{ij}, iqr_{ij}) representing the central tendency and dispersion of each time-slot. At the end of the K-means algorithm, each time-slot is assigned to one of the $K = 2$ clusters. In order to measure the distance between the clusters, we refer to a *cluster separation index* ρ defined as:

$$\rho = \frac{dist\,(c_0, c_1)}{max\,\{dist\,(c_0, origin\,),\, dist\,(c_1, origin\,)\}} \qquad (6)$$

where *dist* is the Euclidean distance operator, c_0 and c_1 are the centroids of the two clusters, and *origin* is the point corresponding to (0,0). The numerator in Eq. 6 represents the inter-cluster distance, whereas the denominator is a normalization factor that makes ρ independent of the

scale of the values, so that $0 < \rho < \sqrt{2}$. Low values of ρ imply that the distribution of the clusters is similar, whereas high values of ρ suggest that the two clusters are well separated. A *similarity threshold* $\overline{\rho}$ must be set to determine if two clusters are distant or could be unified in the same group. In general, we can consider a threshold equal to $\overline{\rho} = \frac{\sqrt{2}}{2} \approx 0.7$, that is in the middle of the interval of ρ. The security analyst could change the threshold in order to adjust its sensitivity in identifying clusters with similar distribution.

In Table 4, we report the results of the K-means applied to the groups considered in Figures 14. In Table 4, the time-slots (from 0 to 23) assigned to the first and second clusters are denoted by *empty cells* or an *x symbol*, respectively.

From these results, we can observe that by considering a similarity threshold $\overline{\rho} = 0.7$, in both trojan wired and trojan external groups, we have that $\rho < 0.7$, thus suggesting that the distribution of the alerts in these groups is similar and could be treated as the same. However, it is interesting to observe that the *K*-means is able to detect that in the trojan external during businessdays, there is a weaker activity between 9am and 4pm (that is possibly related to attacks coming from different time zones). In this example, the smallest value of ρ is represented by the trojan external during holidays, that exhibit a similar distribution in all the time-slots. On the other hand, the trojan wifi alerts are signaled because they form two separate clusters with $\rho \approx 0.95$, and a period $T_i = 24$ that confirms the different behavior of these two groups. In particular, during wifi businessdays, the K-means correctly identifies the daytime time-slots between 8am and 4pm.

9.2. Spatial re-aggregation

The choice of the initial aggregations φ depends on the specific objectives of the security analyst, such as monitoring specific subnets and/or classes of alerts. However, if the descriptive statistics of a specific alerts group A_i are unstable, and if the *temporal re-aggregation* is unable to determine relevant separations δ_i, then it is difficult to determine which anomaly detection approaches could be effective on A_i. In this case, we suggest to perform a *spatial re-aggregation* where the framework refines the alerts group A_i into finer grained aggregations, such as individual hosts.

Spatial re-aggregation is motivated by the fact that the considered alerts series are the result of several contributions related to different hosts and alerts signatures, and depend on several endogenous and exogenous factors that may complicate the detection of robust trends and patterns useful for anomaly detection. An indicator suggesting the need for spatial re-aggregation is represented by the instability of descriptive statistics (see Section 6.3), because if the descriptive statistics related to A_i are unstable, then it is difficult that an anomaly detection algorithm can be effective on ts_i (see Section 7).

By considering series related to finer aggregations, it could be possible to identify whether anomaly detection approaches could work well on novel subgroups. A spatial re-aggregation based on individual hosts is the most intuitive, although other choices are possible, such as individual alerts signatures. This host-based approach produces a group of series in which each element represents the number of alerts generated by each host. By comparing these series, it is possible to identify which are the most active hosts, whether some of them exhibit robust trends and patterns on which anomaly detection approaches can work, whether novel aggregations among similarly behaving hosts may be feasible or convenient.

We describe the spatial re-aggregation by considering as example the trojan wired series considered in the previous sections. We recall that Figure 5(a) shows that in the former period the descriptive statistics of the trojan wired series are unstable, especially in terms of dispersion (*iqr*), and then they are stable in the latter period. We consider a spatial re-aggregation in which we separate individual hosts series. As each host may generate a small number of alerts, it is convenient to adopt a time granularity *g* equal to a day. In Figures 15 we report the host-based alerts series corresponding to the four most active hosts in this aggregation. In each plot, the *X*-axis reports time in terms of days, and the *Y*-axis represents the number of alerts related to each host. The figures are aligned vertically and share the same *X*-axis for a better comparison. Host 1 is the first generating alerts, followed by hosts 2, 3 and 4, respectively. This order is important because further inspections revealed that most alerts are due to a malware (specifically, a fake antivirus software) that propagated from host 1 to hosts 2, 3 and then 4.

From Figures 15, we can observe that hosts 1 and 3 present a continuous and relatively stable generation of alerts, and they are uniformly active during the different time-slots of the day. This might be motivated by the fact that wired clients may be left on 24/7, even during night hours. On the other hand, hosts 2 and 4 exhibit a more sporadic activity, that is probably due to the fact that they are often turned on/off. From an anomaly detection perspective, the most stable results of hosts 1 and 3 indicate that threshold- and distribution-based anomaly detection could be effective on these two hosts, whereas the temporal behavior of hosts 2 and 4 is too noisy and should be studied separately.

This example also evidences an issue that must be addressed if we want to reach an even better effectiveness in the extraction of relevant descriptive statistics for anomaly detection: if we refer to clients and not to 24/7 active servers, we should consider that some hosts may be turned off (e.g., host 2 and host 4), and hence they might not show any alerts activity even for long periods. Hence, further margins of improvement exist for fully automatic data analyses.

10. Conclusions

Modern network environments are extremely dynamic and complex, hence preliminary studies based on exploratory data analysis and descriptive modeling are needed to understand which (semi-)automatic algorithms can highlight relevant security events. We propose the first framework for the investigation of temporal trends and patterns of security alerts related to large networks with the purpose of identifying whether and which anomaly detection approaches can be effective on different alerts groups. We present several applications of the proposed framework by referring to real alerts collected from a large network environment over several months. The results show that, although the alerts exhibit different behaviors and statistics, the framework is able to evaluate the effectiveness of the most popular anomaly detection approaches even in dynamic contexts influenced by many endogenous and exogenous factors that determine high variations in the number and nature of security alerts. As a future research, we are considering the integration of studies on cross-correlation and causality of alerts series, and solutions for online tuning and estimation of parameters for anomaly and system state change detection.

Acknowledgments

We thank the anonymous reviewers for their insightful comments, that guided many modifications in the paper and that helped in clarifying its scope and innovative results.

This research has been funded by the European Commission within the project “SYPCIT - SYstem for Prevention and Combat Identity Theft”, EU Grant Agreement EC DG Home Affairs n. HOME/2013/ISEC/AG/FINEC/4000005234.

References

[1] B. Mukherjee, L. T. Heberlein, K. N. Levitt, Network intrusion detection, Network, IEEE 8 (3) (1994) 26–41.
[2] D. E. Denning, An intrusion-detection model, IEEE Transactions on Software Engineering (2) (1987) 222–232.
[3] M. Roesch, et al., Snort: Lightweight intrusion detection for networks., in: LISA, Vol. 99, 1999, pp. 229–238.
[4] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM Computing Surveys (CSUR) 41 (3) (2009) 15.
[5] J. Viinikka, H. Debar, L. Mé, A. Lehikoinen, M. Tarvainen, Processing intrusion detection alert aggregates with time series modeling, Information Fusion 10 (4) (2009) 312–324.
[6] R. Sommer, V. Paxson, Outside the closed world: On using machine learning for network intrusion detection, in: Security and Privacy (SP), 2010 IEEE Symposium on, IEEE, 2010, pp. 305–316.
[7] D. J. Hand, H. Mannila, P. Smyth, Principles of data mining, MIT press, 2001.
[8] J. W. Tukey, Exploratory data analysis, Vol. 231, 1977.
[9] B. Li, J. Springer, G. Bebis, M. H. Gunes, A survey of network flow applications, Journal of Network and Computer Applications 36 (2) (2013) 567–581.
[10] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, B. Stiller, An overview of ip flow-based intrusion detection, Communications Surveys & Tutorials, IEEE 12 (3) (2010) 343–356.
[11] A. Lakhina, M. Crovella, C. Diot, Diagnosing network-wide traffic anomalies, in: ACM SIGCOMM Computer Communication Review, Vol. 34, ACM, 2004, pp. 219–230.
[12] G. Thatte, U. Mitra, J. Heidemann, Parametric methods for anomaly detection in aggregate traffic, IEEE/ACM Transactions on Networking (TON) 19 (2) (2011) 512–525.
[13] X. Qin, W. Lee, Statistical causality analysis of infosec alert data, in: Recent Advances in Intrusion Detection, Springer, 2003, pp. 73–93.
[14] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, et al., Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation, in: DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings, Vol. 2, IEEE, 2000, pp. 12–26.
[15] J. Viinikka, H. Debar, Monitoring ids background noise using ewma control charts and alert information, in: Recent Advances in Intrusion Detection, Springer, 2004, pp. 166–187.
[16] J. Viinikka, H. Debar, L. Mé, R. Séguier, Time series modeling for ids alert management, in: Proceedings of the 2006 ACM Symposium on Information, computer and communications security, ACM, 2006, pp. 102–113.
[17] F. Valeur, G. Vigna, C. Kruegel, R. A. Kemmerer, Comprehensive approach to intrusion detection alert correlation, IEEE Transactions on Dependable and Secure Computing 1 (3) (2004) 146–169.
[18] H. Debar, A. Wespi, Aggregation and correlation of intrusion-detection alerts, in: Recent Advances in Intrusion Detection, Springer, 2001, pp. 85–103.
[19] R. Sadoddin, A. Ghorbani, Alert correlation survey: framework and techniques, in: Proceedings of the 2006 International Conference on Privacy, Security and Trust: Bridge the Gap Between PST Technologies and Business Services, ACM, 2006, p. 37.
[20] H. Debar, D. A. Curry, B. S. Feinstein, The intrusion detection message exchange format (IDMEF), document type definition.

[21] C. Kruegel, W. K. Robertson, Alert verification determining the success of intrusion attempts., in: Proc. First Workshop the Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA 2004), 2004, pp. 25–38.
[22] C. Kruegel, W. Robertson, G. Vigna, Using alert verification to identify successful intrusion attempts, Praxis der Informationsverarbeitung und Kommunikation 27 (4) (2004) 219–227.
[23] M. Colajanni, M. Marchetti, M. Messori, Selective and early threat detection in large networked systems, in: Proc. of the 10th IEEE International Conference on Computer and Information Technology (CIT 2010), Bradford, UK, 2010.
[24] K. Alsubhi, E. Al-Shaer, R. Boutaba, Alert prioritization in intrusion detection systems, in: Network Operations and Management Symposium, 2008. NOMS 2008. IEEE, IEEE, 2008, pp. 33–40.
[25] K. Julisch, Clustering intrusion detection alarms to support root cause analysis, ACM Transactions on Information and System Security (TISSEC) 6 (4) (2003) 443–471.
[26] S. T. Eckmann, G. Vigna, R. A. Kemmerer, STATL: An attack language for state-based intrusion detection, Journal of Computer Security 10 (1) (2002) 71–103.
[27] M. Marchetti, M. Colajanni, F. Manganiello, Framework and Models for Multistep Attack Detection, International Journal on Security and its Application (IJSIA) 5 (4) (2011) 73–92.
[28] L. Wang, A. Liu, S. Jajodia, Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts, Computer communications 29 (15) (2006) 2917–2933.
[29] R. Shittu, A. Healing, R. Ghanea-Hercock, R. Bloomfield, M. Rajarajan, Intrusion alert prioritisation and attack detection using post-correlation analysis, Computers & Security.
[30] P. J. Brockwell, R. A. Davis, Introduction to time series and forecasting, Vol. 1, Taylor & Francis, 2002.
[31] S. T. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks, Communications Surveys & Tutorials, IEEE 15 (4) (2013) 2046–2069.
[32] C. C. Zou, W. Gong, D. Towsley, Code red worm propagation modeling and analysis, in: Proceedings of the 9th ACM conference on Computer and communications security, ACM, 2002, pp. 138–147.
[33] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, N. Weaver, Inside the slammer worm, IEEE Security & Privacy 1 (4) (2003) 33–39.
[34] Snort users manual, http://manual.snort.org/ (visited in January 2015).
[35] D. Dagon, C. C. Zou, W. Lee, Modeling botnet propagation using time zones., in: NDSS, Vol. 6, 2006, pp. 2–13.
[36] M. G. Kendall, J. K. Ord, Time-series, Vol. 296, Edward Arnold London, 1990.
[37] T. T. Soong, Fundamentals of probability and statistics for engineers, John Wiley & Sons, 2004.
[38] R. L. Eubank, Spline smoothing and nonparametric regression.
[39] D. J. Poirier, Piecewise regression using cubic splines, Journal of the American Statistical Association 68 (343) (1973) 515–524.
[40] D. J. Sheskin, Handbook of parametric and nonparametric statistical procedures, crc Press, 2003.
[41] D. C. Hoaglin, F. Mosteller, J. W. Tukey, Understanding robust and exploratory data analysis, Vol. 3, Wiley New York, 1983.

[42] D. Breitgand, E. Henis, O. Shehory, Automated and adaptive threshold setting: Enabling technology for autonomy and self-management, in: Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on, IEEE, 2005, pp. 204–215.
[43] D. C. Montgomery, E. A. Peck, G. G. Vining, Introduction to linear regression analysis, Vol. 821, John Wiley & Sons, 2012.
[44] N. Ye, S. Vilbert, Q. Chen, Computer intrusion detection through ewma for autocorrelated and uncorrelated data, Reliability, IEEE Transactions on 52 (1) (2003) 75–82.
[45] P. J. Rousseeuw, A. M. Leroy, Robust regression and outlier detection, Vol. 589, John Wiley & Sons, 2005.
[46] A. M. Law, W. D. Kelton, W. D. Kelton, Simulation modeling and analysis, Vol. 2, McGraw-Hill New York, 1991.
[47] G. Tallis, Goodness of fit, Wiley Encyclopedia of Clinical Trials.
[48] M. J. Hinich, Testing for gaussianity and linearity of a stationary time series, Journal of time series analysis 3 (3) (1982) 169–176.
[49] A. Kind, M. P. Stoecklin, X. Dimitropoulos, Histogram-based traffic anomaly detection, Network and Service Management, IEEE Transactions on 6 (2) (2009) 110–121.
[50] E. Parzen, On estimation of a probability density function and mode, The annals of mathematical statistics (1962) 1065–1076.
[51] D. C. Montgomery, Introduction to statistical quality control, John Wiley & Sons, 2007.
[52] M. Basseville, I. V. Nikiforov, et al., Detection of abrupt changes: theory and application, Vol. 104, Prentice Hall Englewood Cliffs, 1993.
[53] A. Bhattacharyya, On a measure of divergence between two multinomial populations, Sankhyā: The Indian Journal of Statistics (1946) 401–406.
[54] F. Yamaguchi, F. Lindner, K. Rieck, Vulnerability extrapolation: Assisted discovery of vulnerabilities using machine learning, in: Proceedings of the 5th USENIX conference on Offensive technologies, USENIX Association, 2011, pp. 13–13.
[55] G. V. Cormack, T. R. Lynam, Online supervised spam filter evaluation, ACM Transactions on Information Systems (TOIS) 25 (3) (2007) 11.
[56] J. A. Hartigan, Clustering algorithms.
[57] D. L. Davies, D. W. Bouldin, A cluster separation measure, Pattern Analysis and Machine Intelligence, IEEE Transactions on (2) (1979) 224–227.
[58] G. W. Milligan, M. C. Cooper, An examination of procedures for determining the number of clusters in a data set, Psychometrika 50 (2) (1985) 159–179.

Figure 1: Framework overview.
Figure 2: Time series per hour of the trojan wired, trojan wifi and trojan external alerts.
Figure 3: Boxplots related to the descriptive statistics for alerts distribution (aggregation $\overline{\delta}_i^1$).
Figure 4: Temporal dependence for trojan-activity wired, wifi and external alerts.

Table 1: Percentage of alerts in the most active classes.

Alerts class [34]	**Percentage**
trojan-activity	83.98%
successful-recon-limited	4.67%
misc-activity	3.04%
bad-unknown	2.96%
attempted-admin	2.48%
misc-attack	2.04%

Table 2: Statistics on coefficient of variation (aggregation $\bar{\delta}_i^1$).

Aggregation		**Coeff. of variation (ν_{ij})**		
		wired	wifi	ext
businessdays	daytime	9.06	1.11	1.56
	night	24.01	3.25	1.11
holidays	daytime	2.42	2.77	1.59
	night	1.35	5.07	1.49

Table 3: Values of stability indexes.

Aggregation		**Stability indexes**	
		β_m	β_{iqr}
wired	daytime	0.10	**0.27**
	night	0.10	0.12
wifi	daytime	**0.49**	**0.31**
	night	0.00	0.12
ext	daytime	0.17	0.12
	night	0.12	0.05

Table 4: K-means results for suggesting temporal re-aggregations.

time-slots		Clustering labels																							T_i	ρ
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23		
wired	bday										x	x		x	x	x	x	x								0	0.50
	hday				x	x																				0	0.43
wifi	bday									x	x	x	x	x	x	x	x	x								24	0.95
	hday														x	x	x									24	0.91
ext	bday								x	x	x	x	x	x	x	x	x									0	0.48
	hday	x	x	x		x	x	x	x						x	x	x	x	x	x	x	x				0	0.27