Detecting Scanners: Empirical Assessment on a 3G Network

Vincenzo Falletta and Fabio Ricciato
(Corresponding author: Vincenzo Falletta)

Forschungszentrum Telekommunikation Wien
Donau-City-Straße 1, A-1220 Vienna, EU
(Email: {falletta;ricciato}@ftw.at)
(Received July 14, 2007; revised and accepted Dec. 5, 2007)

Abstract

Malicious agents like self-propagating worms often rely on port or address scanning to discover new potential victims. The ability to detect active scanners based on passive traffic monitoring is an important prerequisite for taking appropriate countermeasures. In this work we evaluate experimentally two common algorithms for scanner detection based on extensive analysis of real traffic traces from a live 3G mobile network. We observe that in practice a large number of alarms are triggered by legitimate applications like peer-to-peer and suggest a new empirical metric for discriminating between worms and p2p scanners.

Keywords: Anomaly detection, mobile networks, scanning detection

1 Introduction

In networking the term *scanner* refers to an automated program aimed at discovering listening hosts or services within a network. From a security point of view this may indicate a potentially dangerous activity, e.g. a worm trying to propagate. While worms and malware in general are targeted towards terminal hosts, it has been shown that the aggregate traffic generated by such activities can cause problems to the underlying network infrastructure too. As discussed in [16] (see also [25]), this problem is particularly critical in cellular wireless networks, due to the higher functional complexity of such systems. Therefore the ability of detecting worm traffic becomes important for network operators, not only in the perspective of enforcing direct countermeasures - typically blocking individual sources - but also to gauge the overall level of worm activity so as to enforce network configuration that prevent the potential large-scale effects of such traffic.

On the other hand, one should consider that there are also non-malicious purposes for scanning around the network. For example file sharing clients searching for new servers and/or new peers while bootstrapping or refreshing. Peer-to-peer (p2p) architectures require each node to periodically update its cached network map. In these and other cases discovering the network status via scanning is a legitimate procedure, not related to malicious activities nor malware spreading.

On TCP/IP networks we deal with a *portscan* or an *IPscan* (sometimes denoted as *portsweep*) respectively if many connection attempts are made on different ports for the same IP address, or if the same port (or group of ports) is visited for many different IP addresses. If the scan is generated by multiple sources we have a *distributed* scan. Scanners are intrinsically *blind*: their targets can be inactive, not existing or simply not listening on the specific service. Therefore the scan activity often generates a high number of failed connection attempts (openings), and some common detection methods use specifically the count or share of unsuccessful openings as an anomaly indicator. This approach is exposed to collect false positives. For example consider a WEB user that incurs in a network failure along the path to the server: he will likely iterate the page download attempts, thus generating many failed attempts that could be collectively marked as an IPscan in case that the HTML page includes many objects located on different servers.

All Intrusion Detection Systems (IDS) embed modules for scan detection. The implemented algorithms cover a broad range from simple threshold to more sophisticated methods involving data mining [19] or other statistical approaches [4, 9]. However, independently from the complexity of the available detection methodologies, still it is not clear whether any boundary can be drawn between malicious (e.g. worm spreading) and legitimate (e.g. p2p originated) scanners, emphasizing a different behavior of the respective traffic distributions. It would be interesting also to understand whether these two types of scanners can be discriminated by means of simple algorithms, and what can be achieved with currently available tools.

Therefore we are putting on the table two questions: first, is there any evident difference in the behavior of

malicious and legitimate scanners? Second, is there a simple way to separate them?

We choose to deal only with TCP scanners and we focus our attention on two well known detection techniques. The first one is the classic - and mostly used - approach of counting the number of half-open connections generated by a single source towards different destination IPs/ports. Since destinations are randomly chosen within the address space, and the population of active targets is sparse in such space, the majority of half-open connections are expected to fail. Both the detection schemes considered here are based on such basic assumption. The first algorithm is simply a fixed threshold method: It counts the number $n(t)$ of *distinct* connection attempts (i.e. to different {dst_IP,dst_port} pairs) that have failed for each generic source over a time-window of T seconds, and marks the source as scanner if the process $n(t)$ exceeds a fixed threshold N. The second algorithm considered here, called *Threshold Random Walk* (TRW), is more sophisticated: it is based on *sequential hypothesis testing* and was proposed in [11]. A detailed description of TRW will be given later in Section 4.

We have tested both methods on real traces extracted from an operational 3G cellular network within the scope of the DARWIN project [5]. Our goal was to assess the performances and limitations of these methods in separating between worms and p2p scanners. This work required massive efforts in terms of manual inspection of the traces in order to label the reported scanning alarms, i.e. to reveal the "ground truth" in the traffic. As a side value we show how the composition of the anomalous traffic inside the mobile network has changed over one year, from late 2004 to early 2006.

The rest of the paper is organized as follows: Section 2 offers a survey of the related work. Section 3 describes the monitoring setting used to extract the packet traces that are used for evaluation. Section 4 provides a brief description of the two tested algorithms and of the relevant configurations. In Section 5 we present the evaluation results based on the reference dataset and define a new metric, *Significance*, aimed at improving the detection accuracy of malicious scanners. A further comparison with the legacy algorithms via ROC diagrams is presented in Section 6. Additional details about the composition of the anomalous traffic components in the reference dataset are provided in Appendix. Finally in Section 7 we draw the conclusions and outline some directions for progressing the work.

2 Related Work

There are several works focused on the detection of scanning activities (see e.g. [8, 14]). A common goal for most of them is to detect new types of attacks besides the already known ones. In [18] the authors introduce a Worm Detection System (WDS) to monitor a set of hosts inside a local network and to detect infected scanners. Two algorithms are used together to reduce the number of observations for detection and the number of false positives: *Reverse Sequential Hypothesis Testing* and *Credit-Based Connection Rate Limiting*. This solution seems interesting but there are also limitations for this approach which are shown as well as ideas that could be implemented to solve some of these flaws. Another work [22] focuses on detecting scanners inside a backbone. Three different algorithms are compared: the simple one used by SNORT [21] - triggering an alarm if detecting more than N connections in T seconds -, a modified version of Threshold Random Walk (TRW), and a new algorithm based also on sequential hypothesis testing, namely *Time-based Access Pattern Sequential hypothesis testing* (TAPS). It is shown that the latter holds the best performances, but nothing is said about its limitations, despite the presence of false negatives is explicitly noticed. An interesting idea comes from [23] where a "virus throttle" mechanisms is implemented to limit the propagation rate of a malware. With this approach normal traffic remains unaffected, while the virus throttle mechanism is shown to be successful in blocking the spreading of a real worm in less than one second. A wider intent is pursued in [13], by finding a way to classify various kinds of network anomalies from the analysis of the traffic features distribution captured by some metric, denoted as *sample entropy*. By applying the *subspace method* on the traffic multivariate timeseries it is possible to separate them into a regular component plus an anomalous vector. After this separation the characterization is done by grouping together the anomalies that have been detected in timeseries generated at different aggregation levels, and thus a table with all the different kind of anomalies encountered is built. In the end, a strong analytic instrument is used to obtain much more than a simple scan detection, i.e. a clustering of all the anomalies. Using entropy as a metric to detect the anomalies is also done in [10], where a baseline distribution is estimated offline with a recursive procedure and then it is used together with current traffic distribution to calculate the *relative entropy*, which gives the distance between the model and the measured features. The problem of this approach is that one has to feed the detection system with online traffic and periodically update the baseline distribution which has to be calculated offline, but the update procedure has not been implemented yet. The *earlybird* system proposed in [20] claims to automatically detect a new worm and its fingerprint by a so called *content sifting* approach, which is similar to the methodology used to detect the heavy hitters: instead of characterizing a flow by the canonical 5-tuple (*src, dst, srcport, dstport, protocol*) they apply *content fingerprinting* in order to find prevalent payload patterns that could identify a worm signature. The system is shown with all its features and limitations, and some hints are given to use the same methodology to solve other relevant problems like DoS attacks, spam detection, at high data speed and with low memory consumption. Finally, another well known technique for anomaly detection is the

use of *honeypots*. An honeypot is a host (usually virtual) that listens and replies to all unsolicited requests coming from outside, without generating any spontaneous traffic. Honeypots can be considered as a useful complement to the detection methods based on passive capture (see e.g. [15, 24]).

3 Monitoring System and Input Dataset

The present work is based on packet-level traces captured in the operational network of a major mobile provider in Austria, EU. In the following we present a brief description of the network and of the monitoring system used to extract the traces.

The reference network structure is sketched in Figure 1. The Mobile Stations (MS) are connected via radio link to the antennas. In our network four different access schemes are possible depending on the geographical location of the MS and its terminal capabilities: GPRS, EDGE, UMTS and HSDPA [2]. A set of geographically neighboring antennas is connected to a *controller*, called Base Station Controller (BSC) in GPRS/EDGE and Radio Network Controller (RNC) in UMTS/HSDPA. These are then connected to a set of so-called Serving GPRS Support Nodes (SGSN). The overall set of antennas, BSC/RNC and the connections until the SGSNs constitute the so-called Radio Access Network (RAN). The primary role of the SGSN is to perform mobility management function, involving frequent signaling exchanges with the MS. In a typical network there are several SGSNs located at different physical sites. The data-plane traffic collected by the SGSN is concentrated at a small set of so-called Gateway GPRS Support Nodes (GGSN). The GGSN acts as the IP-layer gateway for the user traffic: it is in charge of setting up, maintaining and tearing down a logical connection to each active MS, called "PDP-context", that is conceptually similar to a dial-up connection. During the set up of the PDP-context an IP address is dynamically assigned to the MS. The set of SGSNs and GGSNs are interconnected by a wide-area IP network that will be hereafter referred to as the "Gn network" (ref. Figure 1) following the terminology of 3GPP specifications ("Gn interface"). Across the Gn network the IP packets coming from/directed to each MS are tunnelled into a 3GPP specific protocol (GPRS Tunnelling Protocol, GTP [2]) and then encapsulated into an IP packet travelling between the current pair of SGSN/GGSN. After the GGSN, the user packets enter into the "Gi network" section that is functionally similar to the Point-of-Presence of an Internet Service Provider: it is connected externally to the large Internet and includes internally a number of IP-based service elements, e.g. application servers, WAP gateway, proxies, DNS, firewalls.

For this work we monitored the GGSN links on the Gn interface (ref. Figure 1). All the GGSNs co-located at a single physical site were monitored, corresponding to a fraction x (undisclosed) of the total network traffic[1]. The monitoring system was developed in a previous research project (see [5]). The capture cards are Endace DAG [7] with GPS synchronization. For privacy reasons we store only the packet headers up to the transport layer, i.e. application payload is stripped away. The traces are completely anonymous: any identifier that is directly or indirectly related to the user identity (e.g. IMSI, MSISDN) is hashed by means of a secure non-invertible function. All frames are captured, i.e. sampling is NOT adopted. On the Gn interface the system is capable of parsing the GTP layer and tracking the establishment and release of each PDP-context, hence to uniquely identify the MS associated to each packet, sent or received. Similarly to timestamps, an unique MS identifier - denoted by `MSid` - is stored as an additional label information for each frame. Note that the `MSid` can not be referred to the user identity, it only serves the purpose of enabling packet-to-MS association in post-processing.

For this work we analysed the complete traces from the monitored Gn links over two different days: December 1st, 2004 and April 18th, 2006. Table 1 summarizes the relative proportions of total traffic in the two datasets.

1) *Packet size:*
Our results show that the impact of block cipher mode on the packet size is greater than stream mode; especially as the packet size increases. "Figure **??**" shows the percentage increase in packet size as a function of the original packet size for block cipher mode (top line) and for stream mode (bottom line).

The packet size increase has negative effects not only on the bandwidth usage but it also impacts on the transmission delay, router internal delays, queuing delay, thus affecting jitter and overall packet delay.

2) *Crypto-engine:*
In order to measure the maximum encoding rate, when both algorithms are used, we performed the following experiments. We considered both cryptographic algorithms and for each case we generated 4 packet flows with packets of size 60, 100, 250, 1000 bytes, respectively. Each flow starts from 0 pps and increases its rate of 25 pps every 30 s in order to saturate the crypto-engine.

4 Overview of the Algorithms

Hereby we describe the two detection algorithms that we have tested. In this work we are interested in detecting only TCP scanners (UDP traffic is not considered)

[1]Several quantitative values are considered business sensitive by the operator and subject to strict non-disclosure policy, e.g. absolute traffic volumes, number of active MS, number and capacity of monitored elements. For the same reason some of the following graphs reporting absolute traffic values have been rescaled by an arbitrary undisclosed factor, while distribution graphs have been truncated.

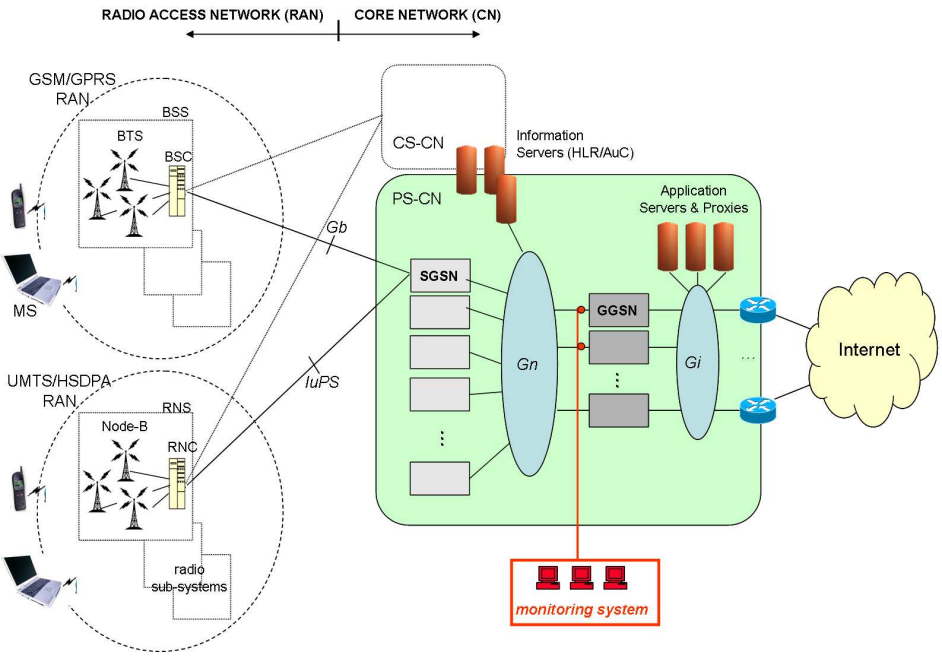


Figure 1: Network structure and monitoring settings

Table 1: Datasets details (absolute values undisclosed)

	Dec. 1st, 2004	Apr. 18th, 2006
# packets processed	tot_pkt_{dec}	$tot\_pkt_{apr} \simeq 5.2 \times tot\_pkt_{dec}$
# bytes processed	tot_byte_{dec}	$tot\_byte_{apr} \simeq 6.9 \times tot\_byte_{dec}$
# MS seen	tot_ms_{dec}	$tot\_ms_{apr} \simeq 4 \times tot\_ms_{dec}$

installed on the MS. Hence, we focus exclusively on the TCP connections attempts travelling uplink, i.e. sent by the MSs. We identify the packet source by the `MSid` field instead of the source IP address: in this way we can unambiguously refer each packet to its source MS even across different PDP-contexts and/or in presence of address spoofing.

The first tested method is the simplest one. It was first proposed in [22] and will be referred to as *syncount*. A scan activity is marked if one source (i.e. a MS in our case) sends more than N SYN packets within T seconds towards different {dst_IP,dst_port} pairs without receiving the corresponding ACKs. More specifically, we have implemented this method in a ultra-simplified version: we roughly count the SYN (in uplink) and SYNACK (in downlink) packets for each MS, without stateful tracking the three-way handshake procedure for each connections. Under the assumption that each active target would respond only with a single SYNACK, such scheme would reveal *how many* - but not *which* ones - connection attempts have succeeded and failed for each MS. The advantage of such approach is clearly to simplify the implementation, as it involves only cumulative counters per each MS but no stateful tracking of individual TCP connections.

The second algorithm is instead considerably more complex to implement. It is known as *Threshold Random Walk* (TRW) and was proposed in [11]. It applies the concept of *sequential hypothesis testing*: for each source r, we define the sequence $Y = \{Y_i\}$ of the connection attempts towards the i^{th} distinct {host,port} pair: we set $Y_i = 0$ in case of success and $Y_i = 1$ in case of failure. Note that here we need to track the state of the three-way handshaking triggered by each SYN packet (this implies some scalability concerns in case of massive SYN flooding). The function $\Lambda(Y)$ is computed as the likelihood ratio:

$$\Lambda(Y) \equiv \frac{\Pr[Y|H_1]}{\Pr[Y|H_0]} = \Pi_{i=1}^{n} \frac{\Pr[Y_i|H_1]}{\Pr[Y_i|H_0]}$$

where H_0 is the null hypothesis that the source r is "benign" (not a scanner). Conversely H_1 represent the hypothesis that source r is a scanner. $\Lambda(Y)$ is updated in real time for each outcome Y_i and it is compared to two thresholds η_0 and η_1 that are set according to some theoretical assumptions (for details see [11]). The hypothesis H_0 or H_1 is accepted if $\Lambda(Y) \leq \eta_0$ or $\Lambda(Y) \geq \eta_1$ respectively.

We have implemented both algorithms in a Linux environment and tested them on two one-day trace datasets: Dec. 1, 2004 and April 18, 2006. From now on, we will refer to the implemented modules as `syncount` and `trw`. For each module we tested two different configurations, referred to as conf#1 and conf#2. All the parameters for

Table 2: Parameters of the two configurations tested - Windowing $T = 60$sec

module	conf#1	conf#2
`syncount`	$N = 50$	$N = 100$
`trw`	$\theta_0 = 0.8$ $\theta_1 = 0.2$ $P_D = 0.99$ $P_F = 0.01$ $\rightarrow \eta_0 \simeq 0.01$ $\rightarrow \eta_1 = 99$	$\theta_0 = 0.6$ $\theta_1 = 0.4$ $P_D = 0.9999$ $P_F = 0.0001$ $\rightarrow \eta_0 \simeq 0.0001$ $\rightarrow \eta_1 = 9999$

all four settings are given in Table 2.

For both methods conf#2 is more conservative than conf#1, i.e. in general conf#1 will raise more false positives than conf#2. For `syncount` this simply means that the alarm threshold N is set higher for conf#2. As for `trw`, we have to set four parameters as input: the desired detection probability P_D, the desired false positive probability P_F, and two conditional probabilities θ_1 and θ_0 defined as $\theta_1 = \Pr[Y_i = 0|H_1]$ and $\theta_0 = \Pr[Y_i = 0|H_0]$. θ_1 and θ_0 represent the probability that the i^{th} distinct connection initiated by a source r was successful, given that the source is respectively a scanner or it is not. By setting these values we can compute the threshold values η_0 and η_1 as $\eta_0 = (1-P_D)/(1-P_F)$ and $\eta_1 = P_D/P_F$ (see [11] for more details). Both `syncount` and `trw` modules use a sliding window T of one minute: at time t, all the pending connections older than $t - T$ are assumed to be failed.

5 Evaluation and Results

5.1 Evaluation of the Algorithms

The results of our runs for both methods are compared in Figure 2. The first observation is that the number of "suspicious" MS that have produced an alarm changes dramatically in the two dataset, increasing by a factor $x = 3.5$ from December 2004 to April 2006 (i.e. $tot\_suspic_{apr} \simeq 3.5 \times tot\_suspic_{dec}$). In both datasets `trw` conf#1 reports the highest number of MS: this variable is taken as the reference for both datasets and the barcharts in Figure 2 have been normalized accordingly. Note that the relative ranking among the four settings w.r.t. the number of alarms is preserved across the two datasets. Note also that the number of alarms relative to `trw` conf#1 has decreased in 2006 for all the other three settings.

Among the suspicious MS there are both true positives (e.g. scanning worms) and false positives (e.g. p2p scanning). In order to classify them properly we had to perform an extensive manual inspection of the traces for each "suspicious" MS. Without the possibility of inspecting the full payload, which is not contained in the traces, the profile of each MS had to be based on the information extracted by packet headers (mainly port numbers)

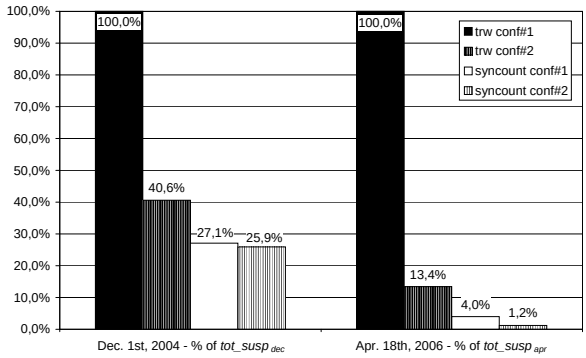


Figure 2: Suspicious MS detected on Dec. 1st, 2004 and Apr. 18th, 2006

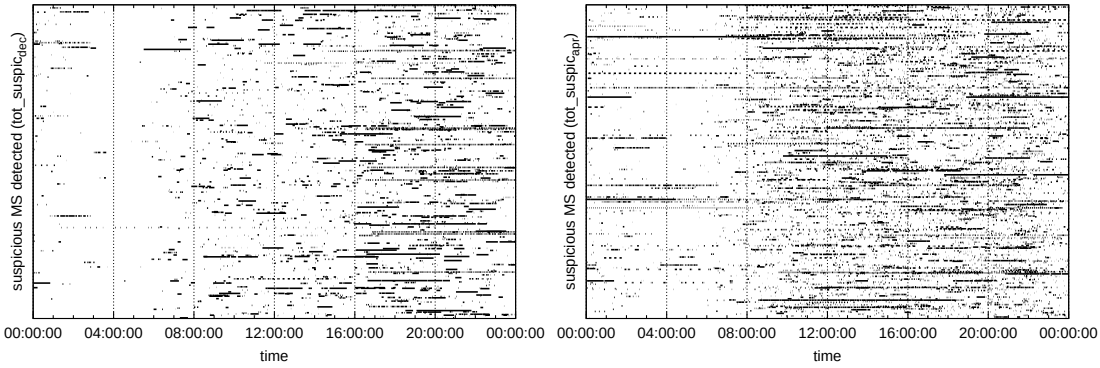
and on the packet timing. We will discuss the results of such analysis more extensively later in Section 6 and provide the complete results in Appendix. For the moment we only highlight in Table 3 the aggregate results for the true positive populations as obtained by manual analysis.

We distinguish three classes: malicious activities ("malware"), p2p applications and other activities not falling in these two classes. As expected, the true positives account for a very small fraction of all MS seen in both datasets (last column of Table 3). However the relative composition has changed dramatically over 1.5 years. In particular, the p2p component has grown conspicuously and became predominant in the 2006 dataset, whereas the malicious component, which was prevalent in 2004, has decreased dramatically in 2006. Such changes are consistent with the following facts. First, following the preliminary analysis of the traces in late 2004 (see [17]), the network operator had adopted some simple measures to counteract the spreading of most common pieces of malware, e.g. by introducing some ad-hoc filtering rules. Furthermore, several users had the time to patch their terminals and remove the infection agent. This explains the sensible reduction in the number of scanning sources from 2004 to 2006 which occurred in our network and, according to a very recent study [1], also worldwide. On the other hand, the popularity of p2p applications among mobile terminals (mainly laptops with 3G cards at that time) has increased, likely driven by the reduction in the mobile tariffs and the general affirmation of some specific p2p applications. In other words, the "usage environment" has changed from 2004 to 2006 in the mobile network under study, also w.r.t. scanning activities, with the relative affirmation of p2p over malware.

As the next step, we were interested in exploring the dynamic behavior of the scanner populations. We resort to the analysis of the time intervals when suspicious activities are revealed. A graphical representation can be easily obtained if we record the timestamps delimiting a suspicious activity, which starts/stops each time the thresholds are crossed. We connect these points drawing an horizontal line to mark each "hot" period. The results for both datasets are shown in Figure 3 (for `trw` conf#1).

Table 3: Composition of True Positives

True Positives on Dec. 1st, 2004				
# of True Positives	Composition			% of tot_ms_{dec}
	% malware	% p2p	% other	
$true\_scanners_{dec}$	66.36%	14.62%	19.03%	1.70%
True Positives on Apr. 18th, 2006				
# of True Positives	Composition			% of tot_ms_{apr}
	% malware	% p2p	% other	
$true\_scanners_{apr}$	7.21%	92.38%	0.27%	0.73%



(a) (b)

Figure 3: Hot intervals detected by `trw` conf#1 on Dec. 1, 2004 (a) and April 18, 2006 (b)

These plots contain the hot intervals relative to tot_suspic_{dec} MS for the 2004 dataset (Figure 3(a)) and to tot_suspic_{apr} MS for the 2006 (Figure 3(b)). Note that during the night hours the overall traffic decreases and so does also the scanning activity. Now for each suspicious MS we calculate the total duration of the correspondent hot activity, thus defining a new variable T_S. We can draw another plot with the cumulative distribution of T_S for all our dry runs, shown in Figure 4.

Each plot in Figure 4 must be interpreted looking at the correspondent histogram bars shown in Figure 2. First we consider Figure 4(a). Each curve represents one of the four runs over the 2004 dataset. We notice a clear correspondence between the curve spacing and the histogram values: that is, the distance between the curves is proportional to the distance between the relative histogram values. The higher difference between two histogram values, the higher distance between the corresponding curves. Besides, we also infer that the runs with lower histogram values have a lower false positive fraction, since CDF curves are shifted down-right, meaning that many MS having low T_S values are neglected. On Figure 4(b) we see that things seem to change for the 2006 dataset. The previously observed relation between histogram values and curve spacing is lost. Indeed, the curves do not appear spaced any more according to the histogram pattern; actually below the curve representing `trw` conf#1 the other curves overlap each other. This difference highlighted in the behavior of the CDF curves is finally what we were searching for. We knew from Table 3 that the 2004 dataset is dominated by malicious scanners. Now we have found that these scanners exhibit a direct proportionality between the duration of suspicious activity and the number of failed connection attempts. This indicates that their scanning rate is somehow constant over time.

On the other hand, for p2p scanners, which are predominant in the 2006 dataset, there is no evidence of strong correlation between the duration and volume of scanning events. This is consistent with the observation that typical p2p applications perform scanning at variable rates. Besides rate regularity, we have found on more point of diversity between the dynamic behavior of the two kinds of scanners, namely the statistics of T_S. It is now evident that relying solely on the count of failed connection, as in `syncount` and `trw`, is not sufficient to discriminate between the two classes. Based on the above findings, we know that T_S could be exploited in the classification method to achieve a better level of separation. In the next section we propose a relatively simple way to achieve that.

5.2 Introducing Significance

As the next step we propose to rank the suspicious activities by a heuristic metric, qualitatively linked to the potential *harmfulness* of the scanning activity itself. For each source r we define its *Significance* (denoted by $\mathbf{S}(r)$)

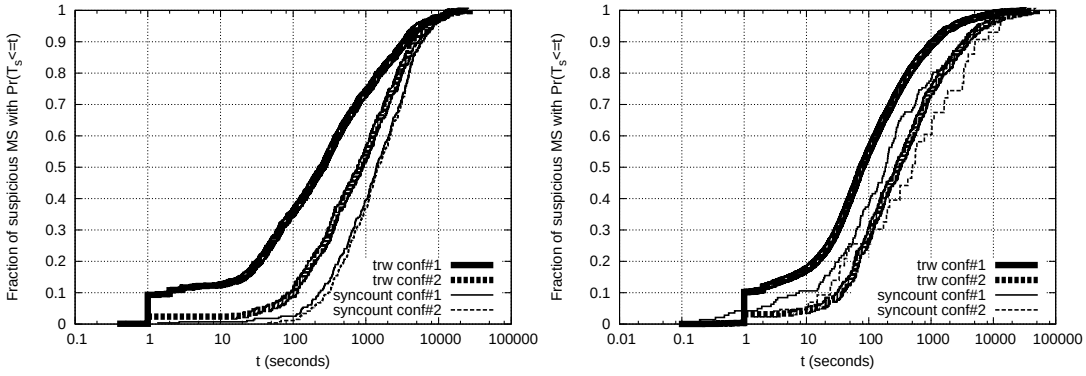


Figure 4: CDFs of T_S - Dec. 1st, 2004 (a) and Apr. 18th, 2006 (b)

as follows:

$$\mathbf{S}(r) = \log\left(N_{tot} \times [1 + N_{tot} - \min(N_{TopPort}, N_{TopIP})]^{\gamma} \times \frac{T_S}{(T_S + T_{idle})}\right)$$

wherein

- N_{tot} is the total number of failed connections initiated by source r;
- $N_{TopPort}$ is the number of failed connections to the top-hitted port by r;
- N_{TopIP} is the number of failed connections to the top-hitted IP address by r;
- T_S is the cumulative total duration of the suspicious activities for r.
- T_{idle} is the cumulated duration of the non-suspicious activities for r, i.e. the total time spent below the alarm thresholds.

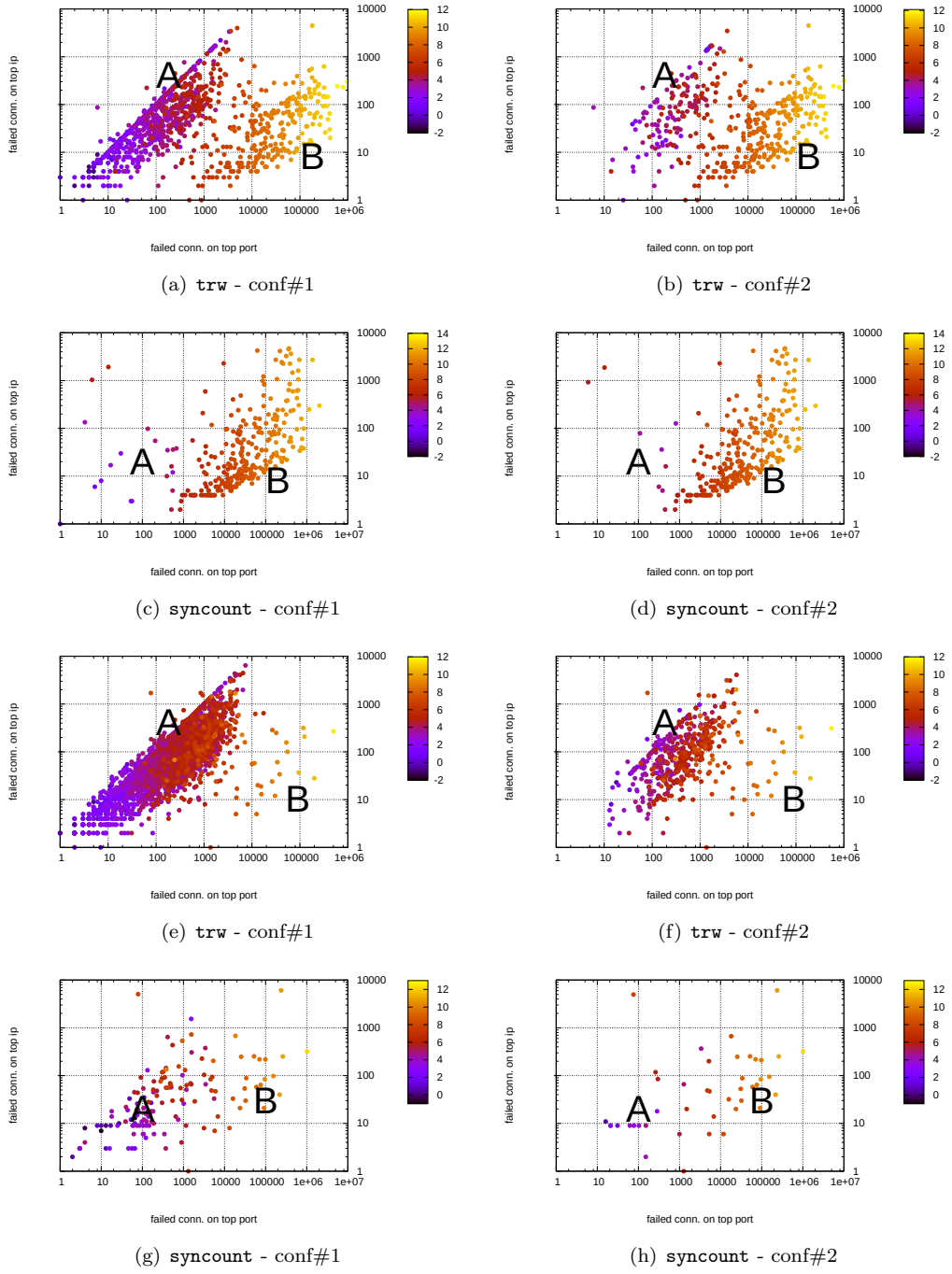
This metric embeds all the key activity parameters encountered so far. It is composed of three factors in a log argument. The first factor is simply N_{tot}. The second factor $[1 + N_{tot} - \min(N_{TopPort}, N_{TopIP})]^{\gamma}$ is the core of the formula: it implements a function that returns an high value if one of the two variables $N_{TopPort}$ or N_{TopIP} is small (or if both are small), the highest value when at least one of them is one. Actually $N_{TopPort}$ and N_{TopIP} can be low at the same time (p2p clients have this behavior). The exponent γ is merely a weight factor, in the following we assume simply $\gamma = 1$. The last factor $T_S/(T_S + T_{idle})$ is simply the ratio between the time spent scanning and the whole time the source was attached to the network: it is a way to account for T_S inside the metric. The product of the three factors is dimensionless and ranges across several orders of magnitude, hence we rescale it by taking the logarithm.

In Figure 5 all the "suspicious" MS are scattered on the plane $\{N_{TopIP}$ vs $N_{TopPort}\}$ for each run, resulting in eight subplots. The value of Significance is encoded by a color map. We clearly see in Figure 5(a) and 5(b) that there are two distinct regions of points: one across the center of the diagrams, and another one on the bottom-right. We label these two clusters A and B.

In Figure 5(c) and 5(d) we see many points in cluster B, but only few ones in cluster A. On the contrary, in Figure 5(e) and 5(f) cluster B has a few points whereas cluster A is well populated. Finally, in Figure 5(g) and 5(h) both clusters A and B have a few points. We claim that cluster A is dominated by p2p flows (false positives) since $N_{TopIP} \simeq N_{TopPort}$, while many IPscanners and other malware are located in cluster B. Indeed in the 2004 dataset the cluster B has an higher point density than in 2006 (compare Figures 5(a)-5(d) with 5(e)-5(h)) confirming the values shown previously in Table 3.

Moreover, the points belonging to cluster B remain constant for both configurations, conf#1 and conf#2, while the number of points in cluster A decreases. This is consistent with the claim that in cluster A we have a large share of false positives, i.e. p2p scanners, which disappear when restricting the detection thresholds (recall that conf#2 is more restrictive than conf#1). Thus, since in Figure 5(c) and 5(d) there are just a few points on cluster A, we infer that `syncount` gets a very low number of false positives.

Remarkably the values of Significance seem to separate correctly the scanning due to malware and to p2p, since the points with the highest $\mathbf{S}(r)$ are always in cluster B. Therefore, we have empirically found a simple method to divide malware and p2p scanners based on a simple scalar metric. We have shown that by choosing a proper dimensional space we could still reduce the classification problem to simple threshold.



(a) `trw` - conf#1 (b) `trw` - conf#2

(c) `syncount` - conf#1 (d) `syncount` - conf#2

(e) `trw` - conf#1 (f) `trw` - conf#2

(g) `syncount` - conf#1 (h) `syncount` - conf#2

Figure 5: Comparison on the plane $\{N_{TopIP}$ vs $N_{TopPort}\}$ of the two datasets: December 2004 (plots a-d) and April 2006 (plots e-h)

6 ROC Analysis of `syncount` and `trw`

The *Receiver Operating Characteristic* (ROC) is a plot of the True Positive Fraction (TPF) versus False Positive Fraction (FPF) and it is a well known technique used to evaluate the quality of a test [6]. To create a ROC diagram one has to proceed as follows. Take the decisional parameter of the evaluation test and rank the population according to the specific value provided by each individual. Then, all these values are ideally assigned to the decision threshold, in turn starting from the highest value down to the lowest one. Using a reference test (assumed to tell the truth) the number of true positives and false positives is calculated and the resulting {TPF,FPF} pair is represented by a single point of the ROC. To obtain the ROC diagrams in our case we need to slightly modify our detection modules to output the value of the decisional parameter for each datapoint (i.e., MS) in order to rank them. For the modified module `trw-ROC` we rank all the

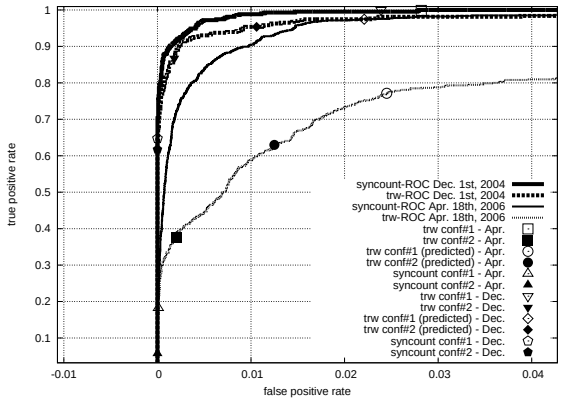


Figure 6: ROC analysis for Dec. 1st, 2004 and Apr. 18th, 2006

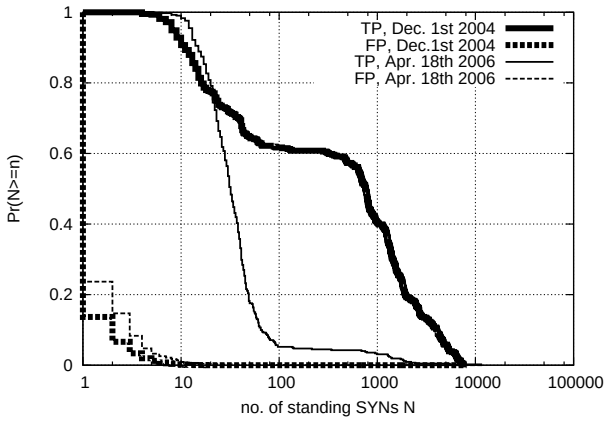


Figure 7: CCDFs of True Positives and False Positives for `syncount`

MS by their Λ_{max} value, while for `syncount-ROC` we do it according to the number of non ACKed SYNs. The result of the manual classification is taken as the "ground truth" reference, as discussed above in Section 5.1. Recall that in this graph we have classified all the kind of scans, malicious and not, as true positives. This has to be remarked because if we had excluded p2p traffic we would have obtained different curves.

We know from the previous section that neither `syncount` nor `trw` are able to discriminate between p2p and malware originated scanning. Besides p2p, there are a number of other phenomena that could generate false positives. For example, a temporary connectivity interruption (e.g. short-term link failure) during a WEB session can cause a burst of new connection openings from the client attempting to reload the page and its objects. These connections fail due to the interruption, but in some cases the client iterates the requests. This can generate a false positive. The reason for comparing the ROC curves for these algorithms is to assess whether the higher implementation complexity of `trw` comes along with any improvement in detection power.

The resulting ROC curves are shown in Figure 6. On both datasets examined, we see clearly that the curve related to `syncount-ROC` has a bigger area compared to the corresponding curve related to `trw-ROC`. Hence, according to the ROC theory, surprisingly `syncount-ROC` has a better behavior than `trw-ROC` in terms of the ratio TPF/FPF. This simply means that the `syncount` algorithm *should* work better, choosing the thresholds properly. Unfortunately, we cannot know in advance the optimal threshold value, which depends on the particular traffic composition - an element that is subject to large changes over time, as we have seen. In the same figure we have highlighted the single points corresponding to our previous runs. For each dataset the points representing `syncount` conf#1 and conf#2 stay below the knee of the corresponding `syncount-ROC` curve, such that FPF$\simeq$0. As regards `trw`, we notice something strange: it seems that there is no matching between the ROC and the single dry runs. Actually conf#2 is identified by points staying almost near the corresponding `trw-ROC` curve, just on the knee, while for conf#1 the points are NOT placed on the corresponding curve but are way detached from it and stay above the knee of the `syncount-ROC` curve with the biggest area, such that TFP$\simeq$1. We must make a brief digression to explain this apparent anomaly affecting `trw` (and not `syncount`). In our implementation of module `trw` the decisional parameter Λ is reset to 1 every time it exceeds the thresholds η_0 or η_1, while in module `trw-ROC` no reset is done since thresholds are not involved; remember that `trw-ROC` simply prints the value of Λ_{max} for each MS so we can rank them and apply the procedure to plot the ROC. Hence, the ROC curve is somehow "dirty" because the reset-to-1 action is not taken in account. Moreover, since the way Λ is updated depends on θ_0 and θ_1, it should be evident that each `trw-ROC` curve in Figure 6 depends on the particular values chosen for θ_0 and θ_1.

To verify the difference between `trw-ROC` predictions and the real dry runs, we highlighted the predicted points corresponding to the two threshold values for conf#1 ($\eta_1 = 99$) and conf#2 ($\eta_1 = 9999$), on both datasets examined. Table 4 reports the differences between the real values of TPF, FPF related to `trw` conf#1 and conf#2 and the highlighted values obtained from the ROC.

On the contrary, for what concerns `syncount`, each point of `syncount-ROC`, correctly predicts the real values of TPF and FPF for all the possible values of the threshold N. Now since we have manually classified all the MS to obtain the above data and plot the ROCs, it can be interesting to plot the cumulative distribution of True Positives and False Positives for `syncount` to find out how to set the threshold N in order to obtain the same TP fraction revealed `trw`.

Figure 7 shows the complementary CDFs for both datasets. This plot contains basically the same information inside the ROC, but in addition the info about N is available. We can see that to obtain the same TPF revealed by `trw` conf#1 (i.e. $\sim$100%) we have to push the threshold down to $N = 4$ for the 2004 dataset and down

Table 4: Fractions of TP, FP, FN and TN for each module, both datasets

	Dec. 1st 2004				Apr. 18th 2006			
	conf#1 ($\eta_1 = 99$)		conf#2 ($\eta_1 = 9999$)		conf#1 ($\eta_1 = 99$)		conf#2 ($\eta_1 = 9999$)	
module	TPF	FPF	TPF	FPF	TPF	FPF	TPF	FPF
`trw`	100%	2.39%	86.77%	0.18%	99.86%	2.82%	40.82%	0.18%
`trw-ROC`	97.45%	2.21%	95.36%	1.06%	77.14%	2.45%	62.99%	1.25%
module	FNF	TNF	FNF	TNF	FNF	TNF	FNF	TNF
`trw`	0%	97.61%	13.23%	99.82%	0.14%	97.18%	59.18%	99.82%
`trw-ROC`	2.55%	97.79%	4.64%	98.94%	22.86%	97.55%	37.01%	98.75%
	conf#1 ($N = 50$)		conf#2 ($N = 100$)		conf#1 ($N = 50$)		conf#2 ($N = 100$)	
module	TPF	FPF	TPF	FPF	TPF	FPF	TPF	FPF
`syncount` (`syncount-ROC`)	64.73%	0%	61.72%	0%	18.23%	0.01%	5.71%	0%
module	FNF	TNF	FNF	TNF	FNF	TNF	FNF	TNF
`syncount` (`syncount-ROC`)	35.27%	100%	38.28%	100%	81.77%	99.99%	94.29%	100%

to $N = 2$ for the 2006. However this would mean paying in terms of FPF, since we would obtain respectively 3.4% of tot_ms_{dec} and 23.7% of tot_ms_{apr}, while `trw` provides 2.39% of tot_ms_{dec} and 2.82% of tot_ms_{apr}. Notice that $N = 10$ seems to be the best choice for both days since we obtain the best compromise between TP and FP. Hence, even if true positive composition changes, this affects the optimal algorithm tuning in a negligible way. Still, all the intrinsic limitations of the methodology do remain.

7 Conclusions

In this work we have tested two common algorithms for scanners detection over two dataset from a live 3G cellular network. The resulting output was then compared against the outcome of extensive manual classification. The two traces refer to distinct one-day periods, more than one year apart (December 2004 and April 2006).

In both datasets the mobile terminal population includes a certain number of p2p users as well as laptops infected by scanning worms, with different relative shares for the two periods. Our main finding is that both tested algorithms systematically report p2p activity as scanners. In other words they do not discriminate between malicious (due to malware spreading) and legitimate (p2p originated) scanning activities. Therefore, they cannot be used to counteract the former without impairing the latter.

Progressing the work, we have proposed an approach based on a novel empirical metric, namely "Significance". In order to discriminate between malware and p2p scanning, the traffic profile by individual terminals need to be represented in a proper space, taking into account the correlation of several features, rather than just relying on the number of failed connection openings. The Significance goes in this direction, and we have shown that it achieves a very good level of separation in our datasets. Comparing the performance of our Significance-based approach versus alternative methods (e.g. the sample entropy was used in [13]) is an interesting direction for future study.

Our experience with real-world traffic and network exposes an additional complication for the task of scanner detection. Most false positives besides p2p are due to maintenance operations (e.g. reboot of internal servers) and other temporary network problems (e.g. local link failures, short-term server failures). Such events, which should be considered physiological in the lifetime of a live network, concur in misleading the detection algorithms which rely solely on the raw counting of failed connection attempts like `TRW` and `syncount`. We have carried out a systematic comparison between these two algorithms based on ROC curves. Our results tell that the two methods yield roughly comparable results, consequently the higher implementation complexity of `TRW` does not seem to be justified. In both cases the actual performances depend greatly on the parameters settings. On the other hand we saw that the optimal setting depends from the actual traffic mix, which is an ever-changing object. This would call for introducing adaptive schemes that are able to automatically tune the sensitivity parameters of the detection engine.

Acknowledgement

This work was supported by the Austrian national funding program Kplus. The views expressed in this paper are those of the authors and do not necessarily reflect the views of the funding partners.

References

[1] M. Allman, V. Paxson, and J. Terrell, "A brief history of scanning," *Proceedings of Internet Measurement Conference (IMC07)*, pp. 77-82, San Diego, California, USA, Oct. 2007.

[2] J. Bannister, P. Mather, and S. Coope, *Convergence Technologies for 3G Networks*, Wiley, 2004.

[3] S. A. Baset and H. Schulzrinne, "An analysis of the Skype peer-to-peer internet telephony protocol," *Proceedings of INFOCOM'06*, pp. 1-11, Barcelona, Spain, Apr. 2006.

[4] D. J. Burroughs, L. F. Wilson, and G. V. Cybenko, "Analysis of distributed intrusion detection systems using Bayesian methods," *Proceedings of IEEE International Performance Computing and Communication Conference*, pp. 329-334, Apr. 2002.

[5] DARWIN home page. http://userver.ftw.at/~ricciato/darwin

[6] J. P. Egan, *Signal Detection Theory and ROC Analysis*, Academic Press, 1975.

[7] *Endace Measurement Systems.* http://www.endace.com

[8] D. Frincke, *Intrusion Detection*, IOS Press, ISBN 1586032542, 2002.

[9] C. Gates, J. J. McNutt, J. B. Kadane, and M. I. Kellner, "Scan detection on very large networks using logistic regression modeling,' *Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC'06)*, pp. 402-408, 2006.

[10] Y. Gu, A. McCallum, and D. Towsley, "Detecting anomalies in network traffic using maximum entropy estimation," *Proceedings of the Internet Measurement Conference (IMC'05)*, pp. 345-350, Berkeley, CA, Oct. 2005.

[11] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 211-225, May 2004.

[12] E. Hasenleithner and F. Ricciato, *Observations at Short Time-Scales from the Edge of a Cellular Data Network*, Technical Report FTW-TR-2007-001, Available online from [5], Jan. 2007.

[13] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," *Proceedings of SIGCOMM' 05*, pp. 217-228, Philadelphia, USA, July 2005.

[14] S. Northcutt and J. Novak, *Network Intrusion Detection.* Sams Publishing, 2002.

[15] G. Portokalidis and H. Bos, "SweetBait: Zero-hour worm detection and containment using low- and high-interaction honeypots, *Computer Networks*, vol. 51, no. 5, pp. 1256-1274, Apr. 2007.

[16] F. Ricciato, "Unwanted traffic in 3G networks," *ACM Computer Communication Review*, vol. 36, no. 2, pp. 53-56, Apr. 2006

[17] F. Ricciato, P. Svoboda, E. Hasenleithner, and W. Fleischer, "On the impact of unwanted traffic onto a 3G network," *Proceedings of 2nd International workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPeru'06)*, pp. 49-56, Lyon, France, June 2006.

[18] S. E. Schechter, J. Jung, and A. W. Berger, "Fast detection of scanning worm infections," *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, pp. 59-81, French Riviera, France, Sep. 2004.

[19] G. Simon, H. Xiong, E. Eilertson, and V. Kumar, "Scan detection: A data mining approach," *Proceedings of 2006 SIAM conference on Data Mining*, pp. 118-129, Apr. 2006.

[20] S. Singh, C. Estan, G. Varghese, and S. Savage, "Automated worm fingerprinting", *Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation*, pp. 45-60, San Francisco, CA, Dec. 2004.

[21] SNORT homepage. http://www.snort.org

[22] A. Sridharan, T. Ye, and S. Bhattacharyya, "Connectionless port scan detection on the backbone," *Proceedings of Malware workshop, held in conjunction with IPCCC*, pp. 567-576, Pheonix, AZ. Apr. 2006.

[23] J. Twycross, and M. M. Williamson, "Implementing and testing a virus throttle", *Proceedings of USENIX Security Symposium*, pp. 285-294, Washington D.C., USA, Aug. 2003.

[24] D. Watson, "Honeynets: A tool for counterintelligence in online security," *Network Security*, vol. 2007, no. 1, pp. 4-8, Jan. 2007.

[25] H. Yang, F. Ricciato, S. Lu, and L. Zhang, "Securing a wireless world," *Proceedings of the IEEE - Special Issue on Cryptography and Security Issues*, vol. 94, no. 2, Feb. 2006.

Appendix A: Anomalous Traffic Classification

In Table 5 we report a list of the top services requested by all the suspicious MS. While in Table 3 we have roughly classified only the True Positives, showing the fractions related to malicious activity, file sharing and other kind of scanners, here we give a more detailed view including False Positives, unknown activities with a low Significance (which are not scanners), as well as true scanners with a high Significance.

Notice again how the weight of "Malware" and "Various p2p" has changed over the two datasets examined. Among p2p programs, it is very interesting to notice the high percentage of Skype in the 2006 dataset, which is not shown in the 2004 because it held for a fraction lower than 0.5%. Yet, we have seen in both datasets many MS attempting connection towards different IPs on ports 80, 443 plus on a random one; the same behavior is observed on Skype users which in addition have a bootstrap signalling activity towards some servers or "supernodes", some of them mentioned in column "Notes" (see also [3]), so we believe that there is a relation between these two entries.

As regards the other entries, most of them represent repeated connection failures to a certain port on few hosts (sometimes on a single host) which result in a low value of Significance, so arguably they are not related to scanners. Another interesting entry is the one related to connection failures to port 80

Table 5: Top services attempted by suspicious MS, fractions of $tot\_suspic_{\{dec,apr\}}$ ($tot\_suspic_{apr} \simeq 3.5 \times tot\_suspic_{dec}$)

Dec. 1st 2004		
% of tot_alerts_{dec}	Service	Notes
39.26%	Various False Positives	
27.21%	IPscan on 135,139,445	Various Malware
7.77%	Connection failures on 80	High Significance
4.37%	6346, 6348, 6349, 4662,1214,random ports	Various p2p
2.72%	8001	Wap, Low Significance
1.94%	7110	Unknown Activity, Low Significance
1.55%	195.230.174.227:1060	
1.55%	8080,8181,8183	Low Significance
1.26%	110	Unknown Activity, Low Significance
1.17%	1.1.1.1:6668	unknown
0.87%	IRC	Low Significance
0.78%	10021	Unknown Activity, Low Significance
0.58%	IPscan on ports 80,443 + random port	High Significance
0.58%	$internal\_proxy_1$:32769	Internal network device
0.58%	81	Unknown Activity, Low Significance
0.58%	5101	Unknown Activity, Low Significance
7.19%	Other	Each entry below 0.5%

Apr. 18th 2006		
% of tot_alerts_{apr}	Service	Notes
50.08%	Various False Positives	
11.06%	6346÷6349; 4661÷4672; 1214; 3531; 2234;	Various p2p
3.56%	Skype (random {*IP,port*})	Signalling on 212.72.49.128 ÷ 212.72.49.159; ports 80,12350,33033,443
3.53%	$internal\_proxy_1$:32769	Internal network device
3.47%	8001	Wap, Low Significance
3.47%	8080,8081,8180,8181,8183	Low Significance
2.58%	IPscan on ports 80,443 + random port	High Significance
1.46%	Random Ports	p2p?
1.43%	81	Unknown Activity, Low Significance
1.37%	8192	Unknown Activity, Low Significance
1.23%	2967	Unknown Activity, Low Significance
0.95%	55111	Unknown Activity, Low Significance
0.64%	9100	Unknown Activity, Low Significance
0.64%	IPscan on 135,139,445	Various malware
0.62%	25	Email virus, High Significance
0.62%	1080	Unknown Activity, Low Significance
0.59%	$internal\_proxy_2$:32769	Internal network device
0.56%	524	Unknown Activity, Low Significance
0.50%	9655	Unknown Activity, Low Significance
0.50%	139.54.61.211:18080	Unknown Activity, Low Significance
11.14%	Other	Each entry below 0.5%

on the 2004 dataset, which disappears in the 2006: this can be explained by the temporary deactivation of an internal proxy for maintenance operations in late 2004.

Vincenzo Falletta is a PhD student in Telecommunications Engineering at Department of Electronics Engineering of University of Roma Tor Vergata, Italy. He received the Laurea degree in Electronic Engineering in November 2003 from University of Palermo, Italy. From November 2003 to November 2004 he was involved in the Italian Research Project FIRB-TANGO, dealing with simulative and analytical models for advanced TCP-IP control techniques, with QoS guarantees. From September 2005 to June 2006 he was researcher at the Forschungszentrum Telekommunikation Wien (FTW) within the scope of the DARWIN project, focused on anomaly detection and performance analysis based on passive traffic monitoring in 3G networks. Since January 2007 he is participating to the IST-DISCREET European project, with the aim of developing innovative solutions for identity management in order to accomplish AAA functionalities in a privacy preserving manner, within a scenario of online service provisioning. He has co-authored 7 publications including international journal papers and conference proceedings.

Fabio Ricciato received the Laurea degree in Electrical Engineering in 1999 and the Dottorato di Ricerca (PhD) in Information and Communications Engineering in 2003, both from the University La Sapienza in

Roma, Italy. During his PhD he collaborated with CoRiTeL, a research consortium led by Ericsson Lab Italy. He participated to the EU project IST-AQUILA and to several national projects on the topic of IP QoS. In 2004 he joined the Forschungszentrum Telekommunikation Wien (FTW) in Vienna, Austria as Senior Researcher and Project Manager for the application-oriented research project METAWIN. The project, focused on traffic monitoring and analysis in an operational 3G mobile network, was run in collaboration with two major industries - a major mobile operator and a system integration company and with the Technical University of Vienna. He was in full charge of the project management and scientific direction. The METAWIN project resulted in the development of an advance monitoring system for GPRS/UMTS networks with high market potential. The prototype is now being consolidated into a carrier-grade product for commercial exploitation. In 2005 he launched the follow-up research project DARWIN, focused on anomaly detection and performance analysis based on passive traffic monitoring in 3G networks. Dr. Ricciato was recently appointed Key Researcher and responsible for the direction of the Packet Networking area of FTW. Dr. Ricciato has co-authored 15 publication on international journal and 30 conference papers. His current research interests are focused on traffic monitoring and analysis, network performance measurements, network anomaly detection, network security.