Design and Implementation of an Anomaly Detection System: an Empirical Approach

Gaia Maselli
Dipartimento di Informatica
University of Pisa
Via Buonarroti 2
56100, Pisa, Italy
maselli@di.unipi.it

Luca Deri
NETikos S.p.A.
Via Matteucci 34/b
56124 Pisa, Italy
deri@ntop.org

Stefano Suin
Centro Serra
University of Pisa
Lungarno Pacinotti 43
56100 Pisa, Italy
stefano@ntop.org

Keywords

Security, network intrusion detection, network monitoring and planning.

Abstract

Network management platforms provide flexible facilities for setting up custom applications able to detect network anomalies on a specific environment. This is because each network is made of users, services and computers with a specific behaviour that is then reflected in the generated network traffic.

Goal of this paper is to show that in every network there are some global variables that can be profitably used for detecting network anomalies, regardless of the type of network users and equipment. As most of the relations among these variables are fixed, this paper shows that it is possible to define generic network rules aimed to automatically detect selected network anomalies. Finally, it covers the design and implementation of an open-source application used to effectively validate this work on a large campus network.

1 Background and Motivation

This paper focuses on network-based intrusion detection and it explores a different approach to the problem. Intrusion detection techniques can be categorised into signature detection and anomaly detection [1][2]. Signature detection systems use patterns of well-known attacks or weak spots of the system to match and identify known intrusions. They perform a pattern matching between network traffic captured and attack signature. If the matching succeds, then the system generates an alarm. The main advantage of signature detection paradigm is that it can accurately and efficiently detect instances of known attacks. The main disadvantage is that it lacks the ability to detect the newly

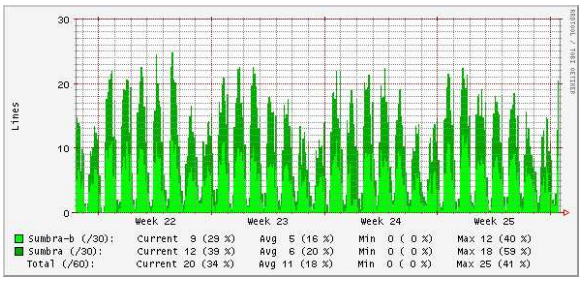

Figure 1: Simple MRTG network traffic curves

invented attacks. Anomaly detection systems flag observed activities that deviate significantly from the established normal usage profiles as anomalies. This paradigm takes the attitude that something that is abnormal is probably suspicious. The construction of such a detector starts by creating a model of what constitutes normal for the observed network, and then deciding on what percentage an activity must be flagged as abnormal. The main advantage of anomaly detection is that it does not require prior knowledge of intrusion and can thus detect new intrusions. The main disadvantage is that it may not be able to describe what an attack is and may have high false positive rate.

The idea behind this work is to define a new type of anomaly detection system, that includes aspects of both signature and anomaly detection techniques. On one hand, it includes a deep study of all well-known attacks that are considered to understand how they influence the normal network behaviour, while they are executed. On the other hand, the defined systems are able to detect not just specific attacks but also unexpected behaviour (anomalies), that can be caused by several kinds of attacks. So, the model that specifies the normal behaviour for the observed network is created by identifying the effects produced over network traffic by an attack and it is independent of the actual network context.

This approach was inspired by some experiments with traffic measurement tools such as Ntop [3], developed for monitoring network traffic, and MRTG [4], a popular tool for network traffic measurement, that confirmed the presence of some similarities on traffic generated by different networks. In fact, although the overall traffic (both in terms of packets and volume) changes significantly, depending on the day and the time of day, the analysis of some popular protocols (e.g. HTTP and SMTP) has produced some traffic distribution curves that do not change significantly over the time.

This fact led the authors to believe that by monitoring some kinds of traffic, it would have been possible, for each network, to draw a specific network traffic curve that does not change significantly over the time, and that could be used to identify network problems (e.g. security flaws). Additional experiments demonstrated that it was quite a difficult task to select those network traffic parameter that allow people to build the network curve of a network. This is because simple curves drawing total traffic that flows across a network interface (see Figure 1), are not very reliable for detecting networks problems, as they can present some peaks (see Figure 2) caused by various reasons that could be a problem (e.g. a DOS attack) or not (e.g. multicast video transmission at the

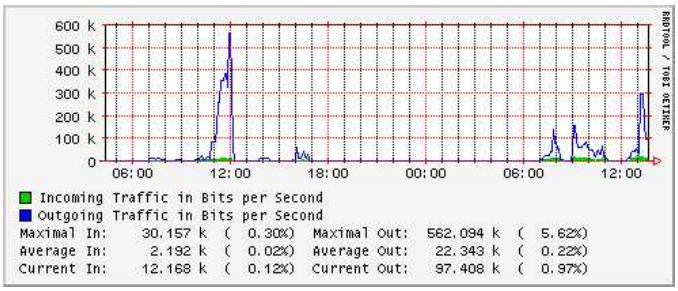

Figure 2: DOS attack effect on network traffic

CS department).

In this case, it is not very wise to use these curves as a litmus paper because the number of false positives could be large, leading the network monitoring system to produce several alarms, most of which need to be discarded. These experiments highlighted that network traffic can show the presence of problems, but it is necessary to identify those variables that can work as network problems detectors. In addition, it is necessary to define some rules that can be profitably used to model network traffic behaviour so that when such rules are violated there is necessarily a network anomaly (e.g. an abnormal network activity).

The following sections cover the outcome of this investigation activity and list the relevant traffic parameters that have been identified as problem detectors. The validation has been performed by implementing a new kind of anomaly detection system (ADS) and by testing it against real network traffic. By anomaly we mean a deviation from the network's expected behaviour that is defined by considering two kinds of knowledge:

- IP protocol specifications contained in RFCs, that needs to be satisfied by every host and network (static knowledge);
- statistical traffic analysis that varies according to network characteristics and type of users (dynamic knowledge).

By referring to both the static and dynamic knowledge it is possible to find out a list of network traffic parameters useful for detecting common network anomalies.

2 Static Network Traffic Knowledge

Static knowledge allows to identify those network traffic parameters that are expected to be verified in each network as they have been defined in standard documents (e.g. RFC) and that are useful for detecting network anomalies. We aimed at finding such parameters by approaching the problem under different points of view:

1. Network Security Violations
   Classify the effects of a security violation over network traffic behaviour for the purpose of identifying a small set of traffic parameters whose value changes significantly during an attack (litmus paper approach).

2. IP Protocol Dissection
   Study the IP RFCs in order to identify those constraints, either mandatory or not, that must always be respected by well-behaved IP applications.

3. Network Traffic Monitoring
   List all the metrics used by traffic monitoring/management RFCs that could be useful for identifying network anomalies.

4. Experience Survey
   Learn what are the traffic parameters monitored by system administrators for rapidly detecting common network problems.

The outcome of this section is a list of network traffic parameters useful for detecting common network anomalies that have been used for the implementation of the tool used to validate this work.

2.1 Network Security Violations

As in real life sickness symptoms are more evident to the doctor when a patient is very hill, the authors decided to study how networks behave under a security attack. In this phase it is not important to know how to detect attack X, but rather to find out how X alters the normal network behaviour. Basically at this stage the authors are not interested in knowing if event A, B and C happen, then somebody is attacking the network using method X, but they want to know what are the network parameters to look for in order to say: "there is something strange out there". In other words the authors want to learn both what network traffic parameters can be used for detecting weird network behaviour and the relations among such parameters in order to detect the problem and raise a security alert. Translating the previous sentence to real life, as doctors know that if body temperature goes above $37°C$ then the patient is sick unless he is running a marathon, in the same way the authors want to find out the measurements (body temperature), the thresholds ($37°C$) and the conditions (unless running a marathon) that must be verified for detecting network sickness .

After having done a survey of several security threats [5], the authors have identified some symptoms and effects that are commons to many attacks, in terms of traffic generated during the execution of the attack. The host executing or suffering the attack can produce these effects. In fact, while some attacks produce strange kinds of traffic that can be recognised by observing network traffic, others use valid packets that cannot be identified. However in the latter case, the victim receiving such packets sends back a sequence of packets we can again recognise. The effects produced on network traffic by the author/victim of an attack, highlight some network traffic parameters useful for detecting anomalies. The following sections list the most common attacks grouped in three families:

1. attacks that exploit IP protocol insecurity;

2. attacks that exploit IP services (e.g. DNS, FTP) insecurity;

3. Denial of Service (DOS) attacks.

2.1.1 IP Protocol Stack Violation

IP protocol insecurity is the ability to exploit the IP protocol lack of specifications or security for the purpose of violating system security [6]. Many network mapping and port scanning techniques are based on sending valid IP packets, passing the checksum control but containing information semantically wrong. Attacks that belong to this family include:

1. Attacks that exploit IP protocol insecurity:
   - Forged packets sent from a spoofed IP source address (e.g. packets with local source address coming from outside the network or packets containing impossible source address).
   - Duplicated or with the same source/destination IP address (land attack).
   - TCP packets sent out of sequence (e.g. TCP packet with the FIN flag set sent over a connection that has not yet been established) or with invalid TCP sequence number for the purpose of terminating or redirecting (hijacking) TCP connections.
2. Attacks that exploit valid IP packets containing unexpected data:
   - Huge packets (e.g. ICMP Ping of Death) that cause remote hosts to hang or crash.
   - TCP packets containing wrong flags combination or out of sequence (e.g. all/no TCP flags set in XMAS/NULL scan, or packets containing the SYN flag and belonging to a session already established, or flags set in other kind of port/network scanning).
   - Packets split in a huge number of fragments or containing invalid IP field values (e.g. invalid IP TTL values for firewalking, i.e. firewall crossing).
3. Attacks that violate the IP protocol specification:
   - TCP three-way handshake exploitation (e.g. incomplete sequence) for the purpose of detecting remote network services (portscan).
   - Invalid (wrong/overlapping/missing) IP fragments.
   - Invalid TCP packet sequence for the purpose of detecting remote host operating system (host fingerprinting).
   - Valid packets sent by the wrong actor (e.g. ICMP Redirect sent by a host that is not a router).

2.1.2 IP Protocol Exploitation

As attacks described so far exploit some IP protocol insecurities, other attacks exploit insecurity of basic IP services for the purpose of disrupting normal network activities. Attacks belonging to this family include:

- Service behaviour change after an attacker gained service root access (e.g. via a buffer overflow).
- Injection of invalid/false/forged information (e.g. ARP poisoning, DNS cache corruption).
- Termination of network services by means of invalid requests that exploit weak service implementations.
- Routing table alteration (e.g. forged ICMP Router Advertising packets).

2.1.3 (D)DOS Attacks

A DOS attack is designed to turn a host/network down by flooding it with useless traffic usually generated by a trojan application. A DDOS (Distributed DOS) instead performs the flooding by sending little traffic from several sources. Attacks belonging to this family include:

- Packets sent from spoofed addressed to invalid destinations (e.g. broadcast/network address) for the purpose of amplifying network traffic (e.g. smurfing).
- Storm of packets (e.g. SYN flood) all sent/originated to/from the same host or network.
- Termination of TCP sessions (e.g. forged TCP RST packet) by exploiting some weaknesses of the TCP/IP protocol.
- Flood storm originated by several different sources and targeted to the same victim.

2.2 IP Protocol Dissection

A classification of the effects produced over network traffic by the previously described attacks, together with a deep study of the IP protocol, has led the authors to identify a subset of anomalous behaviour commons to many attacks. The IP protocol defines some constraints that must always hold. Whenever an application violates them, the protocol specifies how to report the problem (e.g. send an error back to the sender). Unfortunately, the IP protocol is not very strict and it is possible (see section 2.1) that some valid packets are used for nasty purposes. In fact, attacks violate the expected behaviour that is implicitly defined by the IP protocol and that should be satisfied during normal network usage. As IP does not provide any error notification system for these violations, by identifying and defining the expected behaviour, it is possible to detect many attacks as they explicitly violate such behaviour. A group of behaviour corresponds to each of the effects just mentioned:

- Packet semantics behaviour The information contained in packet headers must be consistent with the protocol specification. The IP protocol is quite rich in terms of flags/options that can be specified in packets. Their monitoring and distribution over the time gives an indication about the current network state. For instance, invalid TCP flag combinations can indicate a port scanning, while invalid IP addresses can indicate the presence of a spoofer.

- Activities behaviour

  - Incomplete or incorrect activities
    Any activity that has been started must be completed in the right order. This means that a probe should never report TCP packets sent out of sequence, and that the procedures for establishing/ shutting down a TCP connection should be executed only as specified by the protocol. Dummy activities or messages
  - Dummy activities or messages
    Although not specifically forbidden by IP, each communication must be performed for a certain purpose. Dummy data exchanges usually indicate either a problem or a malicious activity. TCP sessions that have no data exchanged, ICMP fragmented packets, and initial TCP three-way handshake packets with data payload, all belong to this class.
  - Request/response ratio
    Some protocols such as ICMP and ARP define some primitives with 1:1 request/response ratio: zero or one response to a request (e.g. ICMP echo request/response). When the 1:1 ratio is violated this can be the indication of a problem. For instance, in the case of ARP, $if(\#requests) >> (\#responses)$ the sender could perform a network scan for identifying local active hosts or run a misconfigured application that attempts to talk with an inactive host. In the case of ICMP echo, $if(\#responses) >> (\#requests)$ the target machine could be victim of a distributed DOS.

- Traffic distribution behaviour

  - Suspicious Traffic Peaks
    Some kinds of traffic should not be frequent in normal traffic conditions. In fact, besides a number of rare exceptions (e.g. a host that runs a management system), the distribution of some message types (e.g. ICMP Port/Destination Unreachable) or packets (e.g. TCP packet with the RST flag set) is usually very limited and proportional to the traffic generated by a host. This means that peaks or high rates of such messages indicate a problem whose cause needs further investigations. In fact, the peak can be due to a DOS attack, a misconfiguration of a local application (e.g. wrong address of DNS) or an inactivity of a remote application (e.g. POP3 server not active). Average Protocol Distribution

- Average Protocol Distribution
  Statistics have shown that host protocol distribution does not change significantly on a weekly or monthly base. Protocol distribution peaks often indicate network problems or anomalies. Unfortunately in our experience we have noticed that thresholds on protocol distribution are difficult to set as they vary on a host base. On the other hand, when the threshold has been determined, the results in terms of anomaly detection are quite good with a very low rate of false positives.

2.3 Network Traffic Monitoring

After having done a survey about all (over 100) the available network management related RFCs (http:// wwwsnmp.cs.utwente.nl/ietf/rfcs/rfcbytopic.html), the outcome is the following:

1. Most of these RFCs have been designed for monitoring very specific protocols or network media types.

2. The purpose of most of the MIBs is to expose via SNMP statistical, configuration, and accounting information already available on other formats (e.g. the RFC 2925 [7] provides access via SNMP to common operations such as ping and address lookup).

3. Very few RFCs provide information that can be used for detecting network anomalies (e.g. `ipOutNoRoutes` of RFC 1213 [8] and `etherStatsTable` of RFC 2819 [9]).

4. The very few information present in the RFCs that can be useful for our purpose, beside some rare exceptions, it is usually counted per network port or card making it difficult to exploit for precise, per host, problem detection.

5. A few MIBs define alarms and mechanisms for issuing SNMP Traps when a problem occurs, but they rather limit their scope to exposing management information, leaving this task to manager applications.

6. Alarms are usually emitted when a simple threshold, usually over a single variable, is exceeded, whereas network anomalies are usually detected by analysing several parameters with floating thresholds depending on the overall network traffic.

In summary, the survey has been useful for double-checking whether the very few information present in RFCs that can be used for detecting network anomalies was already considered in one of the three other approaches identified by the authors.

2.4 Learning from Experience

Another approach for identifying those network traffic parameters useful for detecting network anomalies, is to interview a group of experienced network administrators and ask them how they identify and tackle problems. The authors have selected different

kinds of administrators ranging from small, single OS (e.g. mostly Apple or Windows) networks to large heterogeneous, multi vendor networks. Below it is summarised the original results of the survey:

1. Control a few core services to control all the services. In IP networks there are some core services (e.g. DNS, routing) that if not fully working cause most of the services to fail. This means that it is very valuable to monitor very well those core services for indirectly monitoring most of the provided network services.

2. The statistical analysis over the time of error responses is usually a good way for detecting anomalies. In fact, depending on the user(s), each computer has a precise usage curve at least with respect to errors. For instance, DNS error responses and ICMP errors are usually very few and their rate is proportional to the overall host traffic. Hosts with over-the-average error rates or errors peaks usually indicate a problem.

3. Non-unicast traffic is often the key traffic. Many protocols (e.g. ARP, OSPF, NetBios, AppleTalk) strongly rely on this kind of traffic; hence the analysis of broadcast and multicast traffic (usually a small portion of the overall traffic) can be quite useful for finding out the overall network status. For instance, misconfigured Windows hosts that have a mispelled workgroup name or mistyped WINS server create a significant amount of broadcast traffic and can be easily detected. In other cases, high ARP traffic rates usually indicate the presence of a local network scanner or proxy ARP probably installed without permission.

4. Precise traffic analysis is very valuable although it is often enough to monitor the overall traffic with the help of some heuristics for obtaining the same result. For instance some peer-to-peer applications are not very secure, hence network administrators discourage their use in some environments (e.g. on the accounting department) as they can cause some serious security threats. In order to precisely identify such applications it would be necessary to implement some decoders that implement stateful inspection, analyse each network flow and learn the type of traffic. Instead, thanks to some heuristics [10], it is possible to avoid implementing decoders and still have an efficient way to detect unwanted network protocols.

5. Active monitoring is often not the most efficient way to check whether a service is available or a client is misconfigured: core services availability can also be monitored by analysing the responses returned by the server to clients. Rejected connection rate can be used for TCP services, whereas ICMP port unreachable can be used for UDP services. Active, but not responding, services can instead be detected by analysing the response vs. request service rate.

6. Experience says that every host has a precise behaviour in terms of protocol distribution. In general such distribution does not change significantly over the time, at least in the short time frame. This means that if a host has never generated IRC (Internet Relay Chat) traffic and at same point in time such host serves IRC

requests, this necessarily means that the host is running either an IRC daemon or a trojan. In conclusion, anomaly detection is not always about detecting unexpected activities but also about detecting state changes.

7. Usually the hosts that are attached first are those that are more vulnerable (e.g. user workstations not maintained by the network administrators), and using those hosts the final attack targets (e.g. the main network servers) are finally reached. For this reason it is important to monitor not only the servers but also the client hosts that are more vulnerable and can potentially produce great problems, as they are physically located on the attacked network.

8. Statistically in the last few years most of the attacks have been performed using a few protocols such as DNS, IRC and FTP. These attacks exploited either protocol or implementation insecurity. For this reasons, hosts that offer services based on the above protocols need to be controlled and managed very precisely as they can be more vulnerable to attacks. This means that alarm threshold values need also to be tuned according to the host vulnerability that is partially determined by the provided services.

3 Dynamic Network Traffic Knowledge

The previous section has shown the result of the investigations carried out for identifying the static network traffic knowledge. This section instead focuses on the dynamic traffic knowledge, based on the idea that it is possible to identify a small set of traffic parameters, useful for detecting network anomalies, analysing traffic statistics. Some parameters are simple counters or gauges that have been selected according to the lessons learnt during the study of static network knowledge, whereas more complex parameters are derived from the composition of simple parameters using simple operators such as ratio or derivative. What leads the authors to label this knowledge 'dynamic' are not the traffic parameters, but the thresholds associated with each parameter, that are not the same for every host, class of user, and network. This is because if host X is used as workstation for reading emails and word processing, its expected traffic curve is not similar to another workstation used for playing multicast videos. Therefore, as each network and host has different thresholds for each of those parameters, it is necessary to have a network information gathering phase that usually takes place as follows:

- network administrators make a quick survey in order to identify all the network assets that need to be monitored (usually the core hosts that run services such as DNS or mail);
- for each asset, the main network protocols (both as client and server) used by the asset are listed;
- a passive traffic monitoring system is configured in order to monitor over the time all the protocols/assets above identified and additional traffic parameters as those identified in the previous section (e.g. the number of SYN packets).

After some time, ranging from a few days up to a month, the gathering phase is over. For each asset, collected information regards a list of all the IP ports on which the host has exchanged traffic as server, and a traffic curve (absolute and derivative value) of each monitored parameter.

The goal of this gathering phase is to produce a traffic model for each monitored asset that includes:

- the list of active services to which remote users can connect;
- some thresholds useful for limiting the bandwidth of potentially dangerous traffic (e.g.SYN packets) or specific protocol traffic (e.g. limit the number of incoming DNS requests);
- a list of traffic parameters that should not be monitored for a given asset as they are not reliable on this specific situation (e.g. a host that runs peer-to-peer applications sends/receives many ICMP errors, hence ICMP cannot be reliably used for such asset as a criteria for emitting alarms);
- a security index that indicates how insecure is the asset, calculated on some parameters including:
  - protocols being used (e.g. a host that accepts telnet connections is more insecure than another based on ssh);
  - kind of users (e.g. asset users that run applications based on weak protocols such as IRC or some peer-to-peer protocols contribute to the overall asset insecurity);
  - service permission (e.g. a host that accepts FTP connections in active mode is more insecure than another host that accepts only passive connections, as the first case is not easy to protect with firewall rules);
  - ICMP "alert" messages (e.g. destination unreachable, source quench, or redirect) emitted/received.

Common thresholds used by the authors include (but are not limited to):

- ICMP echo request/reply ratio, and ARP request/reply ratio useful for detecting whether a host is performing a network scan;
- ICMP Destination Unreachable that may indicate a misconfigured host;
- (# of SYN packets)/(# of active connections) for detecting service scan of DOS attacks;
- asset security index minor than the AS (Autonomous System) allowable security index.

Once the traffic parameters and thresholds have been selected, it is possible to setup:

- an access control list for blocking and logging requests to unwanted protocols;
- a traffic shaper used for attach mitigation and for enforcing some of the above identified thresholds;
- a threshold-based alarming subsystem.

The following section shows in details the validation testbed and describes the selected traffic parameters.

4 Validation: Implementing Network Anomaly Detectors

The previous section highlighted some network traffic parameters used for detecting network anomalies. This section describes the scenario where this work has been validated and it shows how dynamic traffic knowledge parameters have been collected.

In order to validate the work in a real, large network, the authors decided to use the whole network campus of the University of Pisa as testbed.

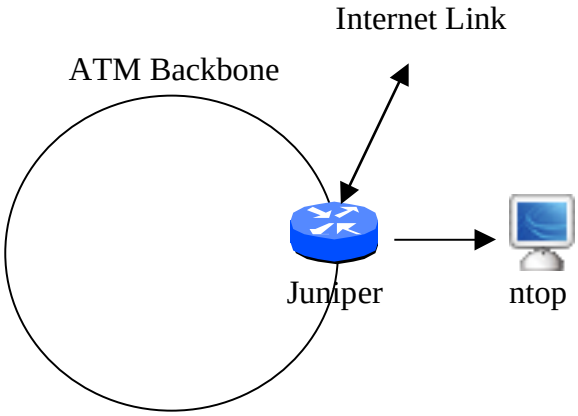

Figure 3: Validation Testbed

The Juniper switch [11] is a high speed switch that sports both ATM and Gigabit Ethernet interfaces. This switch has been configured in a way that the ATM backbone traffic to/from Internet has been mirrored on an Ethernet port where a home-grown traffic probe named Ntop has been installed (see Fig. 3). This configuration allows the probe to see all the traffic from/to the Internet that flows through the 34 Mbit Internet link. Thus, the authors can validate the work on a large network without having to place different probes one in each department under analysis.

The decision to use a Juniper switch instead of a conventional router (e.g. Cisco) has been made for several reason. First, the Juniper box takes care of converting IP-over-ATM traffic into Ethernet packets, so that the probe does not need to have an ATM card for capturing and analysing backbone traffic. Second, it is very fast and allows to test thousand of traffic rules per port with almost no performance degradation. Finally, it has a quite flexible configuration language that enables administrators to define:

- packets/volume counters for traffic that matches some traffic rules (e.g. fragmented HTTP traffic);
- traffic shapers per network flow;
- filters for logging and discarding certain kind of traffic.

Although the Juniper box is very flexible, its measurement capabilities are quite rudimental as they have been designed for measuring overall traffic and not fine-grained host traffic. For this reasons we decided to enhance Ntop to integrate into it the ability to measure all the traffic parameters useful for detecting network anomalies, and to store information into a database for statistical analysis. The implementation of the alarming system and anomaly detector has been realized outside of ntop, in order to avoid creating a large monolithic application difficult to manage and configure.

Figure 4 highlights the current Ntop architecture. The Ntop core is responsible for capturing and analysing network packets. Most of the information is kept in memory with some limited caching on disk for storing data accessed very seldom. For each monitored host, Ntop has a set of counters that keep track of the relevant network activities including (but are not limited to):

- The total traffic (volume and packets sent/received) generated/received by the host classified according to network protocol (IP, IPX, AppleTalk, etc.) and, when applicable, IP protocol (TCP, UDP, ICMP, FTP, HTTP, NFS, etc.).
- TCP session history: source/destination, duration, TCP sliding window size and TTL statistics, retransmitted data, fragmented packets percentage.
- Host used TCP/UDP services, operating system type, and address tracking by means of DHCP monitoring.
- Traffic distribution (local vs. remote traffic), network usage (contacted peers, traffic generated by each running application), overall used bandwidth (actual, peak, and average), local subnet traffic matrix.
- Packets distribution: total number of packets sorted by packet size, unicast vs. multicast vs. broadcast, and IP vs. non-IP traffic.
- Protocol utilisation and distribution according to both protocol and source/ destination (local vs. remote).

In addition, Ntop has been extended with new counters (one for data sent, and one for data received) to detect all the anomalies listed in Table 1.

Whenever a counter goes above a threshold, the counter that keeps track of the problem is incremented, an alarm is emitted, and the packet(s) that triggered the alarm is stored on disk for later in depth analysis.

Ntop-emitted alarms are both stored in a SQL database and used to alert users using several ways including SNMP Traps, GSM SMSs, and instant messengers. Although

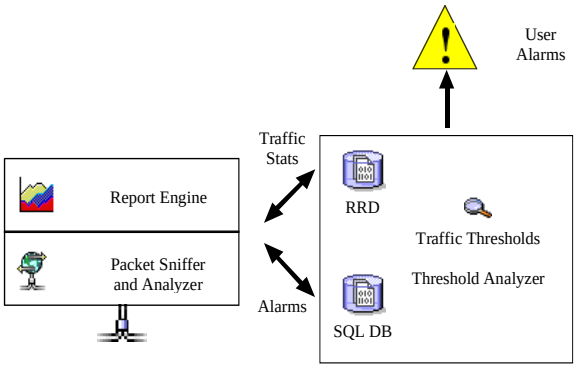

Figure 4: Ntop Security Architecture

TCP Flags	`synPkts, rstPkts, rstAckPkts, synFinPkts, finPushUrgPkts`
Scanning	`ackScan, xmasScan, finScan, nullScan, udpToClosedPort, udptoDiagnosticPort, tcpToDiagnosticPort`
TCP Connections	`rejectedTCPConn, establishedTCPConn, closedEmptyTCPConn, incompleteTWHandshaking`
Fragments	`tinyFragment, icmpFragment, overlappingFragment, icmpFragments`
ICMP	`icmpPortUnreach, icmpHostNetUnreach, icmpProtocolUnreach, icmpAdminProhibited, icmpToBroadcast`
Protocol Checker	`invalidHTTPReq, invalidFTPReq, invalidSMTPReq, invalidSSHReq`
Other	`landAttackPkts, malformedPkts`

Table 1: Ntop Host Security Counters

alarms and counters can partially overlap, the authors have decided to have both for two reason. First, some problems (e.g. portscan or fragmented ICMP packet) are evident after just one alarm, hence the alarming subsystem can alert the user as soon as it sees an alarm without having to wait the polling time. Second, further problems (e.g. network scan) cannot be detected with one single Ntop-generated alarm but with a broader view of the overall network traffic over a specified period of time.

The alarming subsystem is split into two independent components: a traffic information storage, and a traffic analyser. The first one is responsible for periodically polling traffic information (represented in simple ASCII or using high level languages such as XML) out of Ntop via HTTP and storing it on disk. In order to create a modular system the following conventions have been used:

- each counter of each host is stored using the RRDTool [12] on a different RRD (Round Robin Database) file as this format allows to easily maintain large amount of data over the time with limited effort;
- supposing to store the value of counter `tinyFragmentSent` for host Y, the RRD file

that contains the counter is stored on $\$DATA\_DIR/Y/tinyFragmentSent.rrd$.

The traffic analyser is a component written in Perl and responsible for analysing and correlating the data stored in RRD, and generated alarms. The correlation rules used by the traffic analyser are stored on a table inside the same SQL database where the alarms are stored. The format of that table that contains the rules is the following: $< counter\ comparison\ expression > < time\ period > < action >$. For instance:

`‘‘for each host if ((# ARP requests) - (# ARP responses) > 20) over the past 10 minutes then send a trap’’` is translated in one SQL table row:

`(arpSent-arpRcvd)>20`	`10`	`ALARM 'Host $host is sending too many ARP requests: (network scan attempt)'`

`‘‘if host jake sent more that 5 pkts to a closed UDP port in the past 15 minutes then send a trap’’` becomes:

`jake.udpToClosedPort>5`	`15`	`ALARM Host $host sent too packets to a closed UDP port over the past $timePeriod minutes`

where:

- The $< counter\ comparison\ expression >$ is expressed as $< host > . < counter >$: if the host name is not specified then the rule is applied to all stored hosts.
- The variable names start with the dollar sign $ and are expanded by the traffic analyser at runtime for each matching rule.

As the counters rely on RRD, the analyser takes advantage of the facilities offered by RRD for analysing the counter archives and validating the expression: $< counter\ comparison\ expression >$. Nevertheless, it is usually not very wise to emit an alarm every time the analyser detects that a counter is above the specified threshold. In fact, some early tests have shown that it does not make sense to emit an alarm whenever Ntop detects a suspicious event such as data sent to a closed port or a not completed three-way handshake. In addition, in order to detect events such as network mapping it would be necessary to control most of the hosts of the network, making life difficult for network anomaly applications developers. For the above reasons it has been introduced the concept of *risk factor*, an integer value in the range 0-100 that shows how likely the system has detected an anomaly in the network. The risk factor is implemented in the analyser as follows:

- The $< action >$ field can contain the value RISK $< X >$ where $< X >$ is the value of the risk factor.
- Whenever the traffic analysed encounters for a host Y a matching rule containing as RISK action, it increments of X the value of the RRD file $\$DATA\_DIR/Y/-$

$riskFactor.rrd$[1]

- as soon as the value of $\$DATA\_DIR/Y/riskFactor.rrd$ for the current timeframe goes above 100, an alarm is emitted (no additional alarms are emitted if the value is further incremented by additional rules).

As stated before, the Ntop probe has been attached to an Ethernet port of the Juniper box and configured to analyse all the traffic Internet traffic generated/directed to the campus hosts. The alarming subsystem (running on the same host where Ntop is active) stores once every 5 minutes the Ntop traffic counters on disk and validates the traffic correlations rules against the stored traffic.

The main goals of the validation have been to:

- prove that the proposed architecture and implementation can work effectively on a real large network;
- put at work the traffic information gathered by Ntop as show in the previous section by instrumenting the border gateway;
- create a set of correlation rules that allows the campus network administrators to be notified about security violations of campus hosts.

The validation phase lasted more than six months. During the first weeks of validation, the whole system has been strongly modified in order to be able to keep up with the large number of hosts (60 000 hosts) that caused major performance problems. In the remaining weeks, the alarming system has been tuned in order to reduce the number of alarms emitted by the system. It also takes care of hosts that make strong use of peer-to-peer protocols that are characterized by a large number of short living connections and several failed connection attempts.

The set of rules/thresholds/actions produced during the validation phase and the learnt experience has allowed campus network administrators to instrument the Juniper campus border router for:

- blocking the most common attacks;
- detecting common trojans that have been installed on campus hosts
- counting valid yet suspicious traffic (e.g. fragmented UDP traffic originated outside the unipi system);
- adding traffic shaping rules for limiting the bandwidth of some class of traffic (e.g. SYN packets);

[1] When RRD files are created it is specified the measurement interval over which the counter value is computed. Thanks to RRDtool, anomaly detection applications can access historical data for making decisions with no effort as RRDtool stores historical data and performs data reduction automatically each time a new data set is added or modified.

In order to reduce the number of alarms generated by the anomaly detection applications, especially in the case of multiple events received in a small timeframe such as those produced by a network mapping events, the authors experimented several techniques [13][14] including:

- Time over threshold
  In case of alarms emitted for threshold crossing, the alarms are really emitted if an only if, the monitored value persists over the threshold for a specified amount of time.

- Rearm
  When an event triggers and alarms, future events of the same type that would cause the same target to emit the very same alarm are discarded for a specified timeframe.

The above techniques produced good results and allowed anomaly detection applications not to generate frequent alarms for minor problems. On the other hand it is worth to remark that alarm reduction is not a simple task as the experiments shown that almost every network and host need a large amount of time for tuning alarm thresholds and discarding known issues (e.g. a workstation that hosts a management console generates periodically several network mapping alarms). In addition, due to network evolution addition/removal/replacement of computers it is necessary to periodically tune the thresholds according to the new traffic conditions.

5 Evaluation

The way the ADS has been conceived and implemented has several advantages over other similar efforts [5][15]:

1. Anomaly detection based on expected behaviour and the study of RFCs, guarantees a better longevity with respect to detection mechanisms based on pattern matching and signature detection [13]. In the latter case the attack database needs to be updated when a new attack is detected whereas in the case of the ADS there is a good probability to detect new anomalies without having to modify the tool. This means that an ADS can be run mostly unattended with obvious advantages in terms of network management and maintenance.

2. The ADS is effective in many situations where a firewall or an intrusion detection system fail. For instance if an attacker gains root access exploiting a buffer overflow (this is one of the most common attacks) and then takes over a host, the ADS:

   - cannot detect the attack itself as it has been performed using little traffic that it is usually valid from the protocol point of view;
   - can detect the actions performed after the attack (e.g. the attacker installs a trojan that makes some traffic on a TCP port never used before) hence the security violation.

3. This study has highlighted that attacks, when classified in terms of anomaly categories, are very few with respect to the large number of signatures and patterns that similar solutions need to handle. This means that an ADS is much simpler to implement as the information that needs to be handled is very few.

4. The ADS can very well be integrated in an existing network environment. In particular it can feed network appliances such as the latest generation of border routers that allow network administrators to define thousand of network traffic rules per port, mostly produced by the ADS, with no performance glitch.

5. The study of the results produced by the ADS can be very well used for:

   (a) network bandwidth optimisation;

   (b) detection of network bandwidth killers;

   (c) avoidance of unwanted protocols (e.g. printers or proprietary protocols);

   (d) network misconfiguration (e.g. wrong DNS setup, usage of inexistent DHCP/ BOOTP servers); unwanted server activity detection (e.g. installation by mistake of unwanted services);

   (e) TCP/IP stack tuning (very useful for servers) based on the distribution of TCP connection number, flags (e.g. RST, SYN), and latency.

In conclusion, the experiments based on the described architecture have produced good results in terms of detection of network anomalies. As expected, anomaly detection applications are very effective and can run without modifications on different networks. Nevertheless, it is necessary to adjust some thresholds whenever the application is moved as every network/host has its own peculiarities and without threshold tuning the network administrator is often alerted because of issues that are normal on some hosts (e.g. network management stations).

6 Open Issues and Future Work

This paper mostly deals with anomalies detectable at the IP protocol level. The authors are aware that this work should be extended up to include the application level (e.g. DNS). On the other hand, the work described in this paper still remains valid as several application anomalies can still to be detected at IP level, and also because it would be rather costly and challenging to implement a new version of Ntop that contains decoders for most of the protocols. Another work item is the implementation of a user-friendly interface for host-based threshold tuning and alarms discard. In fact the available applications are written in pure Perl applications with limited configurability if not by modifying the source code or tweaking a few configuration files. This is an important extension as when several hosts are monitored it is necessary to keep several alarm/threshold configurations one per (class of) host, which can be difficult to maintain without a user interface. Finally, it is necessary to rework the event notification system, as the current implementation is very simple to use but not very efficient and simple to

keep consistent in terms of configuration with the anomaly detection applications when the network administrator changes some configuration files.

7 Final Remarks

This paper has demonstrated that it is possible to define generic network traffic rules that allow network anomalies to be detected regardless of the host and network type. An application, released under GNU GPL, and based on the assertions described in the previous sections has been used for validating this work using real network traffic. The outcome of this paper is not a firewall-like application but a new application type named anomaly detection system that is quite close to a network management tool.

Such a system highlights network anomalies, regardless of the network type. It requires a limited confoguration and can be easily integrated with a firewall, or a border switch, for enforcing the overall network security.

8 Availability

Ntop, all the code and applications described in this paper are distributed under the GPL 2 licence and can be downloaded free of charge from both the Ntop home page (http://www.ntop.org/) and other mirrors on the Internet. Some Unix distributions including but not limited to *BSD and Linux, come with Ntop preinstalled.

References

[1] K. Jackson, *Intrusion Detection Systems (IDS): Product Survey*, Los Alamos National Laboratory, LA-UR-99-3883, 1999.

[2] H. Debar, M. Dacier, and A. Wespi, *Towards a Taxonomy of Intrusion Detection Systems*, Computer Networks, 31(8):805-822, April 1999.

[3] L. Deri, R.Carbone, and S.Suin, *Monitoring Networks Using Ntop*, Proc. of IM 2001, Seattle, May 2001.

[4] T. Oetiker and A. Van den Bogaerdt, *RRDtool Tutorial*, http://rrdtool.eu.org/, 2000.

[5] S. Northcutt, and J. Novak, *Network Intrusion Detection: An Analist s Handbook*, 2nd Edition, 2000.

[6] S. Kumar, *Classification and Detection of Computer Intrusions*, PhD Thesis, Purdue University, August 1995.

[7] K. White, *Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations*, RFC 2925, September 2000.

[8] K. McCloghrie, M. Rose, *Management Information Base for Network Management of TCP/IP-based internets:MIB-II*, RFC 1213, March 1991.

[9] S. Waldbusser, *Remote Network Monitoring Management Information Base*, RFC 2819, May 2000.

[10] D. Plonka, *FlowScan: A Network Traffic Flow Reporting and Visualization Tool*, Proc. of XIVth Lisa Conference, December 2000.

[11] J. Doyle, *Juniper Network Routers: the Complete Reference*, Osborne, 2002.

[12] T. Oetiker and A. Van den Bogaerdt, *RRDtool Tutorial*, http://rrdtool.eu.org/, 2000.

[13] M. Sylor and L. Meng, *Using Time Over Threshold to Reduce Noise in Performance and Fault Management Systems*, Boston University, 2001.

[14] R. Vilalta, S. Ma, and J. Hellerstein, *Rule Induction of Computer Events*, IBM T.J. Watson Research Centre, 2001

[15] S. Axelsson, *Intrusion Detection Systems: A Survey and Taxonomy*, Chalmers University, March 2000.