Technologies, Methodologies and Challenges in Network Intrusion Detection and Prevention Systems

Nicoleta STANCIU
Bucharest University of Economic Studies, Romania
nicoleta_stanciu@yahoo.com

This paper presents an overview of the technologies and the methodologies used in Network Intrusion Detection and Prevention Systems (NIDPS). Intrusion Detection and Prevention System (IDPS) technologies are differentiated by types of events that IDPSs can recognize, by types of devices that IDPSs monitor and by activity. NIDPSs monitor and analyze the streams of network packets in order to detect security incidents. The main methodology used by NIDPSs is protocol analysis. Protocol analysis requires good knowledge of the theory of the main protocols, their definition, how each protocol works.
Keywords: *Intrusion Detection and Prevention System, Protocol Analysis, Sensor, Signature, State*

1 Introduction

Increasing size and complexity of the Internet and Intranet networks have led to increasing number of vulnerabilities that could be exploited. Thus, the internal and external attacks on the information systems are increasing at an alarming rate. Also, these are becoming more severe and sophisticated. The attackers find ingenious ways to bypass the security controls and to compromise the security and the well functioning of the information systems. They are motivated by financial, political, and military objectives. In this context, defending wide area networks from malicious traffic, unauthorized access to systems involves many problems.

In security information systems Network Intrusion Detection and Prevention Systems (NIDPS) are important tools to detect possible incidents and also, to attempt to stop them in real time. Due to changing attacks, intrusion detection methodologies and technologies continuously evolve, adding new detection capabilities, to avoid detection. They must adapt to new forms of malware, to the public networks, increased traffic.

2 Concepts of Intrusion Detection

An intrusion is a successful action to gain access to an information system, to compromise it or to make it unavailable. This is possible due to the presence of vulnerability in the target system that can be exploited by a motivated intruder.

Intrusion Detection and Prevention is the process of monitoring the information systems by sensors or agents and analyzing the collected information to detect and to attempt to stop the attacks in real time, identifying vulnerabilities, the violation of security policies or standard security practices.

An Intrusion Detection and Prevention System (IDPS) is a tool that monitors information systems, collects, analyzes information, and initiates responses when an intrusion is detected.

Intrusion Detection Systems (IDSs) mainly work as defensive mechanisms. They only alert the system administrators that an incident has occurred. Intrusion Prevention Systems (IPSs) can take some actions to attempt to stop the attack, such as breaking the connection or modifying the firewall rules to deny access to the intruder. The response of the classic IDS can be slow if the system administrator is busy while the response of the IPS is automatic. An architecture that uses together IPS and IDS technologies is the best solution for defense in depth.

Conceptually, a generic IDPS consists of modular components. It mainly has the following components: monitoring system, storage, analyzer, and responder.

- Monitoring system – monitors and logs

the events in a computer system or network;
- Storage – stores information, called audit record, about suspicious activities or intrusions; also, the security policies used in analysis are stored;
- Analyzer – uses different analysis methodologies to detect the incidents;
- Responder – the response mechanism of incidents.

The IDPSs could be classified as:
- By detection methodology [12], [18]:
  - misuse-based detection
  - anomaly-based detection
  - stateful protocol analysis
- By activity [12]:
  - network-based
  - wireless-based
  - network behavior analysis
  - host-based
- By behavior on detection:
  - passive
  - active
- By collection and analysis frequency:
  - continuous
  - periodic

The detection methodologies describe the characteristics of the analyzer.

Misuse-based detection [18] represents known attacks in the form of a pattern or a signature. The main issues in misuse detection methodologies are how to make a signature that encompasses all possible variations of an attack, and that do not also match normal behavior.

Misuse-based detection can be implemented by the following techniques [18]:
- rule-based intrusion detection – the attacks are represented as rules of if-then form;
- model-based reasoning system [18] – the attack scenarios are stored in a database; the *anticipator* searches attack scenarios in audit trail and generates the next set of hypothesized behaviors, that it passes to the *planner*; the planner determines the likelihood of occurrence them in the audit trail; if the likelihood is high the scenarios accumulate;
- state transition analysis –the attacks are represented as a sequence of state transitions, from initial state to compromised state, of the monitored system;
- key stroke monitoring – an attack is identified by user key strokes registration;
- pattern matching model – the signatures of known intrusions are represented as patterns that are compared with audit trail. This approach considers intrusion signatures – patterns, audit trails – abstract event streams, detector – pattern matching.

Anomaly-based detection considers that the intrusive activities are anomalous. This is the process [12] of comparing the profiles of normal behavior against real activity of the system to identify significant deviations. The profiles are developed by monitoring the real activity of users, hosts, networks or applications over a period of time, called a training period, and preservation of what is considered without intrusion. The profiles can be static or dynamic.

Stateful protocol analysis uses protocol model, the IDS sensors perform full protocol decoding for some application-layer protocols. The process [12] compares profiles of normal protocol activity for each protocol state against observed events in the system to identify deviations. The "stateful" [12] means that the IDPS can understand and can track the state of network, transport and application a protocols.

There are four main groups of IDPS technologies [12]:

Network-Based [12] - monitors network traffic for network segments or devices (e.g. packets captured by network interface in promiscuous mode) and analyze the network, transport and application protocol activity to identify possible attacks originating from outside or inside of the system.

Wireless [12] which monitors wireless network traffic and analyzes its wireless networking protocols to identify attacks.

Network Behavior Analysis (NBA) [12] which examines network traffic to identify unusual traffic flows.

Host-Based is installed locally on host machine and monitors the characteristics of the host and events occurring with that host. It analyzes network packets entering and leaving the host, log files on the host, processes running on the host, attempts to execute malicious code. It checks the integrity of system files, files access and modification, CPU usage. By the type of audit data they analyze, there are operating system–level intrusion detection systems and application-level intrusion detection systems.

The first three are network intrusion detection technologies. Network-based is older while wireless and network behavior analysis are newer and have been developed due the increasing complexity of networks.

3 Network Intrusion Detection

Primary source of a Network Intrusion Detection and Prevention System (NIDPS) is network traffic. In the network traffic the data is passed through the layers from source to destination. The four TCP/IP layer are: hardware layer, internet protocol (IP) layer, transport layer, application layer.

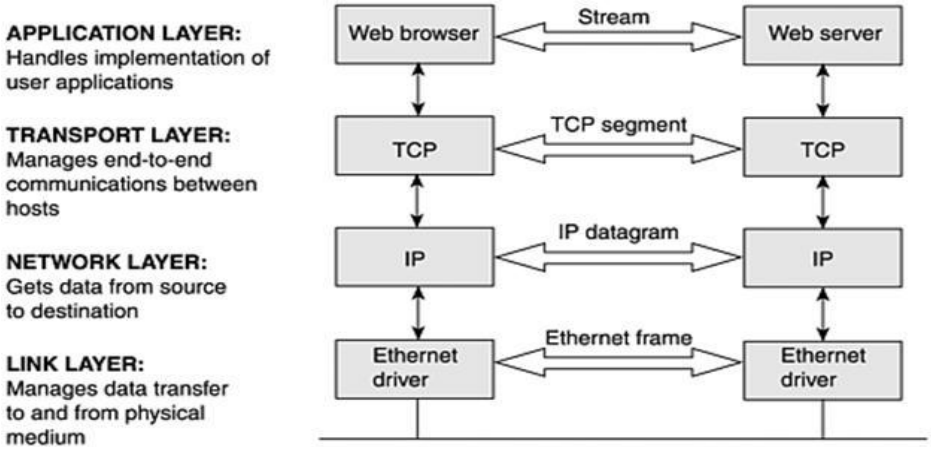

Fig. 1. The TCP/IP Model; Source: [19]

A typical component NIDPS [12] is composed of sensors, one or more management servers, multiple consoles and optionally one or more database servers.

Sensors – monitor and analyze the activity. The sensor can be an appliance-based – a specialized hardware and sensor software or software only. An appliance-based sensor includes specialized NICs and NIC drivers and specialized processors that assist in analysis.

Sensors can be deployed in the following modes [12]:

- Inline – network traffic can pass directly through a NIDPS – Figure 2. This is by definition active as it can inspect every network packet and react in real time on dangerous activities, e.g. dynamically block network traffic that it believes to be malicious. Some inline sensors can be hybrid firewall/IDPS devices but can be specific IDPS.

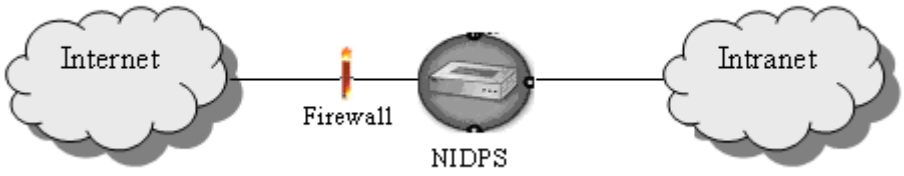

Fig. 2. Inline NIDS

- Passive – monitors a copy of the actual network traffic – Fig. 3. It monitors traffic using a network tap or spanning port [12].
- Network Tap (Test Access Port) – is a direct connection between a sensor and the physical network media itself, such as a fiber optic or copper cable. Fiber Taps [16] split the network signal into two streams, enabling to the network and monitoring devices to receive the signal. The signal must be regenerated to have adequate strength.
- Spanning port [12] – which is a port of a switch that can see all network traffic going through it. If a switch is configured or reconfigured incorrectly, is under heavy loads, its spanning port might not be able to see all traffic.

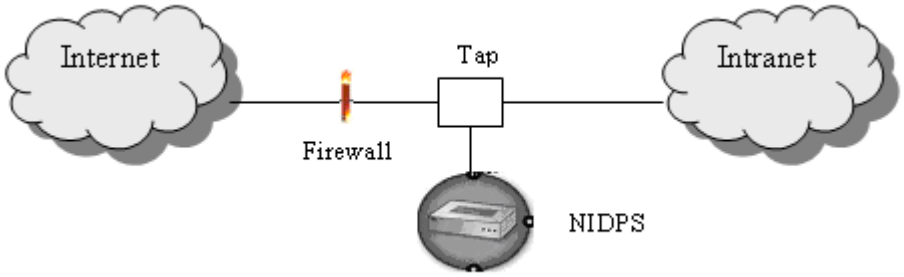

Fig. 3. Passive NIDS

Generally, intrusion prevention techniques require that the sensors be deployed inline mode because the passive sensors monitor a copy of traffic and cannot easily break the connection. They still can place packets onto network in order to disrupt network connection but such method is more cumbersome and less effective. Administrators must decide where the IDPS sensors should be located consistent with security needs.
Most NIDPSs mainly rely on protocol analysis. The types of attacks detected are [12]:

- **network layer attacks** – spoofed IP address, illegal IP header length. The IP, ICMP, IGMP protocols are analyzed;
- **transport layer attacks** – port scanning, unusual packet fragmentation, SYN floods. The TCP and UDP protocols are analyzed;
- **application layer attacks** – buffer overflows, format string attacks, malware transmission. Mainly, these protocols: DNS, FTP, HTTP, IMAP, IRC, POP, SMTP are analyzed;
- **policy violation** – use of inappropriate Web sites or use of forbidden application protocols.

Network-based IDPSs [12] cannot detect attacks within encrypted network traffic, as virtual private network (VPN) connections, HTTP over SSL (HTTPS), and SSH sessions. The analysis must be performed on payloads within encrypted network traffic, thus IDPSs analyze the payloads before it is encrypted or after it is decrypted. However, some IDPSs can also monitor encrypted communications to identify known vulnerabilities or misconfiguration.
Network-based IDPSs [12] may be unable to perform full analysis under high loads, especially if stateful protocol analysis methods are in use. To prevent its disability it uses high-bandwidth network cards, limits the number of simultaneous connections, sets timeouts to expire connection state.
Also, various types of attacks, such as distributed denial of service (DDoS) attacks, and anomalous activity can attempt to exhaust a IDPS sensor's resources and to make them unavailable.
The first methodology was the development of **simple signatures** [13]**,** patterns to be

searched in traffic. In the initial concept, **string matching**, each signature is written for key phrases or commands associated with a known attack. It creates a list of signatures. An incoming packet [13] is compared, byte by byte, with each signature for particular characteristic of malicious traffic, and when there is a match, an alert is generated. Then the next packet is read into memory and the process begins again.

Another concept is **protocol analysis.** In **"protocol analysis"** [9] the IDS sensor uses definition of protocols and understands how various protocols work. At each layer of TCP/IP model [19], the packet consists of a header of its own and data, sometimes known as the payload.

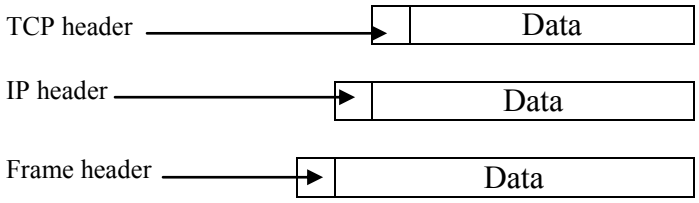

Fig. 4. Packet; Source: [19]

There are IDS signatures that focus on IP, TCP, UDP and application layer protocol **header value** [8]**.** Any header value can be used in signatures, but the most commonly used header-related signature elements are [8]:

- source and destination IP addresses (particularly reserved, non-routable, and broadcast addresses)
- port numbers in TCP or UDP protocols (port scanning attacks)
- header length
- unusual packet fragmentation
- particular TCP flag combinations in TCP headers
- the protocol field in IP headers (enables to distinguish among TCP scans, UDP scans and ICMP scans, SYN flooding attacks and UDP flooding attacks)
- checksum
- Time to Live (TTL)
- ICMP types/codes that should not normally be seen

There are some of the header values clearly abnormal, so they make great candidates for signatures. Classic examples are:

- TCP packet with the SYN and FIN flags simultaneously set[7];
- TCP packet with the SYN, FIN and PUSH flags simultaneously set [19]; It is anomalous because a SYN flag starts a connection, a FIN flag closes a connection and PUSH flag sends data while a connection is opened;
- no TCP flags [19]– if the TCP flag byte field has a value of 00. A byte TCP flag byte field is represented as two hexadecimal characters or nibbles. The high-order nibble contains two of reserved bits for ECN (RFC 3168) and the bit settings for URG, ACK flags. The low-order nibble contains the bit settings for the PSH, RST, SYN and FIN flags. 00 means that no TCP flags have been set. A normal TCP flag byte has at least one flag bit set;

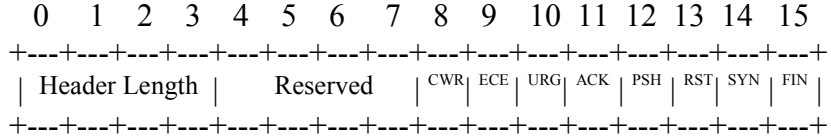

Fig. 5. Header Length and TCP flags – TCP segment; Source: [19]

- TCP flag byte field with a value greater than 3 indicates that one or both of the reserved bits are set (ACK=1=> 2^0=1, URG=1=> 2^1=2 => 1+2=3). Any value greater than 3 in high-order nibble is anomalous unless ECN is being used [19]. ECN is a technique for reducing congestion in a network. ECN traffic should have a non-zero value in the differentiated services byte (formerly known as the type of service byte);
- a bad TCP header length [19] is when the specified TCP header length is greater than the actual TCP segment (header and data) length. The value of the TCP segment length can compute by subtracts the IP header length from the IP datagram total length;
- ACK flag isn't set and the acknowledgment number has non-zero value [7];
- URG flag isn't set and urgent pointer field has non zero values [7];
- the normal IP header with no options is 20 bytes (IP v.4), or five 32-bit words. An IP header that might contain a dangerous IP option such as source routing would have a length of greater than 5 found in this field [19];
- unknown IP protocol number in IP header [22];
- connection attempt from a reserved IP address; it checks the source address field in an IP header [7];
- traffic sent to broadcast address from outside network [19]. The broadcast address has a final octet 255 or 0. The destination address field is found in bytes 16 through 19 (32 bits) of the IP header, so the byte 19 of the IP header must be different from 0 or 255. For example, a malicious host sends many ICMP echo requests with a spoofed source IP (IP address of the victim host/network) to a broadcast address of an intermediate network. The intermediate network must allow inbound broadcast traffic. All the live hosts in the intermediate network send ICMP echo reply to the victim host, because they believe it's the sender. If the intermediate network has many hosts and/or the target host has a slow Internet connection, can occur a denial of service attack on target host;
- the own network's MTU (maximum transmission unit) is smaller than the size of the IP datagram and DF (Don't Fragment) flag is set [19] (to discover the MTU some hosts send across the network a datagram with the DF flag set, and the MTU of the network that required fragmentation is contained in the ICMP error message);
- malicious fragmentation [19]. Fragmentation provides a field of action for attackers, them using to mask and facilitate their exploits. Malicious fragmentation occurs in many different forms. It uses malicious fragmentation to exhaust system resource in some kind of denial-of-service attacks, degradation of service or disabling of the target host, to evade detection or circumvent the monitoring and filtering devices incapable of fragments reassembly. It requires good knowledge of the fragmentation theory to detect malicious fragmentation and recognize normal fragmentation. IDPSs detect and analyze fragmented traffic and discover malicious fragmentation [19]:
  - fragmentation the 20-bytes TCP header (the normal TCP header with no option has 20 bytes) in multiple fragments in an attempt to avoid detection;
  - creation of the fragments with overlapping offset fields – exploits weaknesses in the reassembly process of fragments; When these fragments are reassembled at the destination host some systems will crash, hang or reboot;
  - the length of the last IP fragment was changed [22];
  - a large number of IP fragments can lead to denial of service [22];
  - while not illegal, IP fragments smaller than 500 bytes are unusual [22];

- can cause a denial-of-service by repeatedly sending a non-zero offset fragment to a host.

- the source and destination ports are set to 21 (FTP servers). In normal FTP traffic, it sets a high port number (greater than 1023) as the source and port 21 as the destination [7].

Because ICMP and UDP [8] protocols are connectionless it checks each packet. The TCP protocol is connection-oriented. In this case [8], address and port are constant in all packets in the connection and they can be checked once, but TCP flags should be different among the packets in the session, so it will check every packet.

A header-based signature could include any one or more characteristics. The simple signatures are more prone to false positives while the more complex signatures are prone to false negatives. An example: two or more characteristics can occur separately in legitimate traffic but combined in same packet are very low.

It can create a signature set based on known exploit programs or known and potential vulnerabilities. The signature set based on known exploits has the disadvantage that will be a significant delay between the time the exploit occurs and the time the IDPS can recognize its activity. This signature set is written after the exploit has become public. A signature set based on protocol analysis has the advantage of looking for any signs of abnormal or suspicious activity by checking various fields for abnormal values. Abnormal values for fields protocols can be used only in the presence of existing vulnerabilities. By using the protocol analysis techniques there will be much better detection of known and unknown attacks, it will be more difficult for attackers to evade through change to exploits' code or NIDPS obfuscation.

Above there is a **static analysis**. For better performance **dynamic protocol analysis** [8], [20] is required. TCP, UDP and ICMP headers and payloads are contained inside the payload of IP packets. In order to get TCP header data, for example, it must parse the IP payload. Other protocols such as FTP, DNS, HTTP, SMTP, IMAP, POP3 are contained inside payload of UDP or TCP packets. In this case, it must parse two levels, IP and UDP or TCP, in order to get to them. For this there are analyzer trees [20]. For each connection the system identifies the protocol used and activates the appropriate analyzer. Each intermediate node receives data, analyzes it and passes the transformed data to the appropriate analyzer. By a dynamic processing it can add, change or remove the analysis component.

A superior and flexible NIDPS should [20]:

- use multiple ways to recognize protocols;
- enable multiple protocol analyzers to work in parallel;
- choose the appropriate protocol analyzer in incorrect classification cases;
- can dynamically decapsulate tunnels;
- enable high-speed analysis by performance.

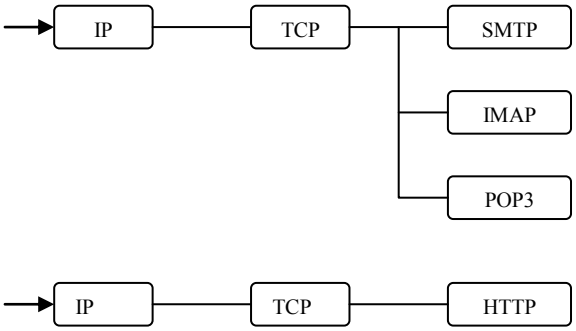

Fig. 6. Dynamic protocol analysis; Source: [20]

Only more advanced NIDPSs perform full protocol decoding, protocol analysis requires much more advanced IDPS sensors capabilities than the simple signature technique. Protocol analysis techniques monitor traffic, recognize a particular protocol performing full decoding, validate it and alert when traffic does not meet expectations. Protocol analysis techniques examine the header values and payload values.

For example [9], protocol analysis to identify an attempt to exploit **buffer overflow** vulnerability in FTP MKD command, verifies the length (that it isn't overly long) and the content (that doesn't contain shell

code) of the argument of the MKD command.

A simple NIDS evasion method is **Path Obfuscation** [10] – alters the path so, it has different appearance but the same meaning. The advanced NIDPSs protocol analysis based detect and stop these types of attack because they perform much of the same processing as a Web server, FTP server or operating system.

For example [10]:

- character escaping; So, id and i\d have the same meaning;
- using excessive whitespace, including TAB and new line. If an attacker creates a SQL injection attempt using DROP TABLE, the NIDPS should ignore the additional spaces;
- using the backslash instead of slash in URL should be treated as slash;
- single-dot sequence – when ./ combination is used in a path, it does not change the meaning. So, the NIDP treats the windows/./system32 as windows/system32;
- path transversal – such as /../ . So, if the attacker uses windows/sample/../system32, NIDPS will wipe out sample, considering windows/system32;
- hex encoding – so, %20 is the hex encoding equivalent of a space, %5c is the hex encoding equivalent of backslash, %2e is the hex encoding equivalent of dot. %5c%2e%2e is path transversal;
- unicode – so (HTML entity), @ is @ and dot is . name1@domain1.com is name1@domain1.com.

Thus, to circumvent the attack attempts described above the advanced NIDSs protocol analysis based perform these [10]:

- examine IP packet header to find IP protocol number. IP protocol number 6 corresponds to TCP protocol;
- examine TCP packet header to find TCP destination port number. If it is port 80, this indicates that the user is sending an HTTP request to the server;
- perform HTTP protocol analysis parsing the HTTP request all component, including the URL's path;
- process the URL path by handling path obfuscation, hex encoding, double hex encoding, or unicode;
- generate an alert if an attack attempt is found.

The evasion methods can be combined to create an advanced evasion technique. Thus, two or more evasion techniques of different network layers can be combined. Two ways of advanced evasion techniques are metamorphic and polymorphic malware [15]. In both cases the code is different and more sophisticated with each iteration to avoid detection. The polymorphic malware code has two parts; one part remains constant with each iteration. For example, if viruses, a virus have a virus decryption routine (VDR) and an encrypted virus program body (EVB). In this case, it is easier to provide a complex signature to identify the constant part. The metamorphic malware is more difficult to detect. For its detection advanced techniques [15] are used, such as generic decryption scanning, negative heuristic analysis, emulation and access to virtualization technologies.

In order to detect the attacks, a traffic normalizer [14] should be placed in path of traffic and to normalize the packet stream. The normalizer should remove the potential ambiguities. Thus, the NIDPSs monitor normalized traffic.

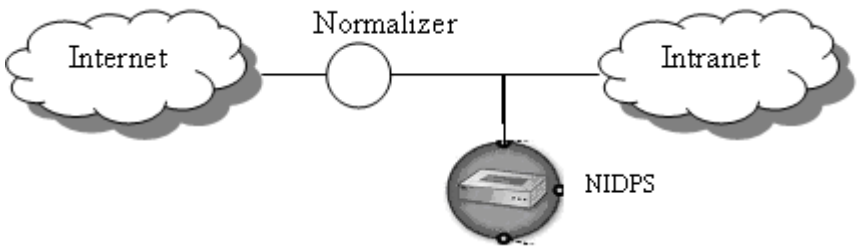

Fig. 7. Typical locations of normalizer and NIDS. Source [14]

By protocol analysis-based as superior intrusion detection solution the packets are examined in detail, using the protocol definitions and making the same processing as a Web server, FTP server or operating system. By this method a much wider range of attacks can be detected, including known and unknown attacks.
Web or FTP servers usually run on well-known port numbers. In static application-layer protocol analysis standard port numbers for protocols are used. But, there are Web or FTP servers that run on other ports with benign or malicious intent, and also, non Web servers run on 80/tcp in order to evade security monitoring. The attackers [20] use application protocols on non-standard ports or on ports assigned to other protocols: Trojans that use non-standard ports; botnets use the IRC protocol on ports other than 666x/tcp; hidden FTP servers for file-distribution on ports other than 21/tcp.
Therefore, a dynamic protocol analysis approach [20] examines a per-connection data structure to identify what analysis to perform for the flow. For example [20], if the destination port for a TCP SYN packet is 80, the NIDS should perform IP, TCP, and HTTP analysis for all packets of the flow. If the payload of a packet on port 80/tcp - initially analyzed as HTTP - looks like an IRC session, it replaces the HTTP analysis with IRC analysis.
To identify whether traffic on standard ports uses the appropriate protocols [20] the NIDS should examine traffic in-depth, by decapsulating tunnels. There are few systems that can perform this [20]. Such a system is [20] McAfee's IntruShield. For example, this can unwrap the SSL-layer of HTTPS connections.
It presents below an example of HTTP signature (Bro) [20]:

```
signature http_server {                    # Server-side signature
ip-proto == tcp                                # Examine TCP packets.
payload /^HTTP\/[0-9]/                     # Look for server response.
tcp-state responder                        # Match responder-side of conn.
requires-reverse-signature http_client     # Require client-side sign. as well
enable "http"                              # Enable analyzer upon match.
}
signature http_client {                    # Client-side signature
ip-proto == tcp                            # Examine TCP packets.
payload /^[[:space:]]*GET[[:space:]]*/     # Look for requests [simplified]
tcp-state originator                       # Match originator-side of conn.
}
```

Another example is a signature (Snort) for a Telnet login failure [13]:

```
     alert tcp $HOME_NET 23 ->
$EXTERNAL_NET any (msg:"TELNET Bad
Login";
     content: "Login failed"; nocase;
flow: from_server, established;
classtype:bad-unknown; sid:492; rev:5;)
```

Thus, the analysis engine searches "Login failed" string in the payload and if this is found an alert is generated.
In **"protocol analysis"** [9] NIDPS sensors perform full protocol decoding for application layer protocols, such as DNS, FTP, HTTP, SMTP. Thus, they have the ability to detect both known and unknown types of attacks.
In the **stateful protocol analysis** [11] approach the NIDPS sensor monitors and analyzes all of the events for the duration of a session and adds stateful characteristics to the protocol analysis. It records information about the connection state. The NIDPS performs correlations among the events occurred and the state of the network, among different events over a connection. Thus, the sensors can detect attacks that cannot be recognized by another way.
Common types of state are [17]:

- **connection state**: for every active connection, the NIDPS sensor records information like duration, status of the handshake, and payload volume;

- **per-host state** - such as connection attempts from a source address to detect scanners;
- **signature state** - for signatures that so far have only partially matched.

There are two new approaches for application-layer protocols analysis [20]:

- statistical analysis of the traffic within a connection – uses an analysis of interpacket delays and packet size distribution to distinguish interactive applications like chat and file transfer, distinguish Web-chat from regular Web. For this, it uses statistical analysis, machine learning components, decision trees or neural networks;
- locating protocol-specific byte patterns in the connection's payload or signatures that can be used to determine components of an HTTP request or an IRC login sequence.

One of the simplest ways to use state in application-layer protocols analysis is to **associate every response with the request that generated it** [11] over a connection. The NIDPS sensors that use stateful protocol analysis for detection can do this.

An example is the server's response of a FTP command [11]. At an attempt to access a FTP server it returns a numeric code that indicates the status of the response. A *2xx* FTP status code indicates that the command has successfully completed, while a *5xx* FTP status code indicates that the command was not successful, and the error is permanent. Thus, it can recognize brute force attacks, by identifying many failed requests in a session. 2xx status code in the response shows that an attacker attempt was successful.

Another example of **state is the phases of a session** [11]. The phases of an FTP session are [11]: connection, authentication, transaction and disconnection. Unauthenticated users should only perform providing usernames and passwords. If the user has authenticated successfully, the session is in authenticated state and the user can perform specific commands, such as, change directory, list the contents of the directory, delete files, delete a directory, make a new directory, copy files. If these commands are performed in the unauthenticated state it can be an attack.

Because the deep packets inspection (the header and the payload) is hard or even impossible, the **flow-based intrusion detection** is a current option studied [2]. With such approach, the communication patterns within the network are analyzed, instead of the contents of individual packets. The flow-based intrusion detection uses flows for input data, instead of packets. A flow [4] is defined as a set of IP packets passing an observation point in the network during a certain time interval. A TCP flow corresponds to a single network connection, while a UDP flow is a stream of packets terminated by an inactivity period. This information is in the form of Netflow [3] or IPFIX [4].

The flow is mainly characterized by [2], [3], [4]:

- source and destination IP address;
- source and destination port number for TCP and UDP;
- protocol field of IP header.

Also, the following parameters are important:

- Type of Service (Diffserv, ECN) value;
- TCP flags of TCP headers;
- packets size;
- flow size.

The **flow-based detection** should be combined with **packets inspection** in detection process [2], [5], [6]. At the first stage flow-based can be used to detect certain attacks. At the second stage, packet inspection can be used for suspicious activities previously discovered. This combined technique is used especially for the analysis of high-speed networks. It applies to DoS, scan, worm, spam, botnet detection.

Accounting flows is a two-step process [2]: flow exporting and flow collection. These tasks are performed by two components: flow exporter and flow collector. The flow exporter, also known as observation point creates flow records from observed traffic. The flow collector retrieves the flows created by the flow exporter and stores them in a form suitable for further monitoring or

analysis. The analysis of exported flow data for intrusion detection can be decomposed into three principal steps [6]:

- flow data is received from the monitoring devices and decoded;
- the flow data is normalized and preprocessed in order to provide appropriate input to the detection algorithm;
- applies a detection algorithm in order to discover network intrusions.

The detection algorithms can be [6]:

- threshold-based - that uses predefined or adaptive thresholds for specific measures;
- principal component classifiers (PCC) – the set of flows are decomposed into their components and the algorithms detects anomalies in multivariate time-series;
- outlier detection algorithms – uses a set of normal data to the learned normal behavior; an outlier is a data point which is very different from a normal data;
- rule learning algorithms - that learn classification rules from training data containing, labeled normal and attack data.

A **flow-based method** of detection is *subspace method*, detailed in [1]. With this method the traffic flows (IP flow) are aggregated at the Origin-Destination (OD) level. It uses samples of flow data from every router. Sampling is random, capturing 1% of packets entering every router. Sample packets are characterized by 5-tuple, IP address and port number for both source and destination, and protocol type. In each sampled IP flow it is also recorded the number of bytes and packets. The OD flow can be represented as a sum of normal and anomalous components, $x=\hat{x} + \tilde{x}$. It examines three distinct representations of sampled flow traffic, as time series of bytes, packets and IP flow, all indexed by the 5-tuple headers. Each anomaly results in a value of the $||\tilde{x}||^2$ that exceeds the threshold statistic. The set of anomalies is cast as triples of *(traffic type, time, OD flow),* where "traffic type" is one of Bytes (B), Packets (P), or IP-Flows (F). It aggregates all triples with the same time value, placing some triples into the new categories BP, BF, FP, and BFP. Thus, a BP anomaly is one that is detected in both byte and packet time series at the same time. It groups triples to form anomalies in space (all OD flows corresponding to the same traffic type and time) and time (all triples with consecutive time values, having the same traffic type). Finally, a set of anomalies results. Each anomaly is due to a set of anomalous OD flows. Thus, it detects the network-wide traffic anomalies, by aggregating sampled flow measurements at the origin-destination level.

The paper [21] proposes a combination of **timeslot-based** and **flow-based analyses** in network anomaly detection.

A first approach is a combined method using the timeslot-based and flow-based in parallel. Network traffic is inputted to both detectors and analyzed by each detector. Because, a large buffer storage in a flow-based analysis represents a problem, to reduce the amount of data to be analyzed by flow-based analysis, a packet of sampling and setting short timeouts was made. The method has the disadvantage that it may result a lack of the information needed to detect anomalous flows. To avoid this, timeslot-based analyses have been proposed in the first stage and flow-based analyses in the second stage. The timeslot-based detection has two modules, header-based detection module and payload-based detection module. Also, in timeslot-based detection, firstly, each slot is classified based on a threshold *(Th_{ac}),* into anomalous slot candidate and normal slot. For normal slots the detector does not transmit anything. By another threshold *(Th_{as}),* the anomalous slot candidates are classified into anomalous slots and suspicious slots. For anomalous slots, the timeslot-based detector triggers alerts. For suspicious slots, in a second stage (flow-based analysis) is performed a detailed analysis.

4 Conclusion

There are many ways to achieve network security and NIDPS are a complement to them. Good knowledge of the networks, how

the protocols work, network threats and vulnerabilities lead to a strong defense in depth. So when it makes a mistake or gets sloppy, it leaves a hole that attackers find and exploit. NIDPS must recognize attacks so that their exploitation can be prevented.

Good knowledge of methods and technologies incorporated into every product leads to a good choice of products implemented since each product has its own detection capabilities and every computer system has specific threats and vulnerabilities. Depending on the degree of appropriateness between the informatic system and the NIDPSs, a more or less effective and complete activity of a monitoring and control results.

All the methodologies combine in modern products, exploit inherent strengths of each approach and prevent the weakness from leading to a superior product.

Network intrusion detection systems have a number of fundamental limitations. Many systems have a very high false positive rate, they are vulnerable of evasion attacks, denial-of-service attacks. Therefore they must be improved. They must adapt to new types of attacks, to achieve the security and protection of networks and computer infrastructures. It is clear that using dynamic protocol analysis increases the number of security breaches that can be detected.

References

[1] A. Lakhina, M.Crovella, C. Diot, "Characterization of Network-Wide Anomalies in Traffic Flows", *IMC'04 Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, ISBN:1-58113-821-0, pp. 201-206, 2004

[2] A. Sperotto et al., "An Overview of IP Flow-based Intrusion Detection, Communications Surveys & Tutorials, IEEE, Vol.12, pp. 343-356, 2010

[3] B. Claise, "Cisco Systems NetFlow Services Export Version 9", *RFC 3954,* 2004

[4] B. Claise, "Requirements for IP Flow Information Export (IPFIX)", *RFC 3917,* 2004

[5] C. Rossow et al., "Sandnet: Network Traffic Analysis of Malicious Software", *BADGERS '11Proceedings of the First Workshop on Building Analysis Dataset and Gathering Experience Return for Security,* ISBN:978-1-4503-0768-0, pp. 78-88, 2011

[6] G. Münz, G. Carle, "Real-time Analysis of Flow Data for Network Attack Detection", *Integrated Network Management,* 2007

[7] K. Kent, Network Intrusion Detection Signatures, Part One, 2001, available on-line at www.symantec.com/ connect/articles/network-intrusion-detection-signatures-part-one

[8] K. Kent, Network Intrusion Detection Signatures, Part Two, 2002, available on-line at www.symantec.com/ connect/articles/network-intrusion-detection-signatures-part-two

[9] K. Kent, Network Intrusion Detection Signatures, Part Three, 2002, available on-line at http://www.symantec.com/ connect/articles/network-intrusion-detection-signatures-part-three

[10] K. Kent, Network Intrusion Detection Signatures, Part Four, 2002, available on-line at http://www.symantec.com/ connect/articles/network-intrusion-detection-signatures-part-four

[11] K. Kent, Network Intrusion Detection Signatures, Part Five, 2002, available on-line at www.symantec.com/ connect/articles/network-intrusion-detection-signatures-part-five

[12] K. Scarfone, Peter Mell, Guide to Intrusion Detection and Prevention Systems, National Institute of Standards and Technology, 2007

[13] M. Tanase, The Great IDS Debate: Signature Analysis Versus Protocol Analysis, 2003

[14] M. Handley and V. Paxson, "Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics ", *Proceedings of the 10th USENIX Security Symposium,* 2001

[15] M. Rouse, "Metamorphic and Polymorphic Malware", 2010, available

on-line at http://searchsecurity.techtarget.com/definition/metamorphic-and-polymorphic-malware
[16] NetOptics, Deploying Network Taps with Intrusion Detection Systems, available on-line at www.netoptics.com/products/pdf/Taps-and-IDSs.pdf
[17] R. Sommer, Viable Network Intrusion Detection in High-Performance Environments, 2005
[18] S. Kumar, E. Spafford, A Pattern Matching Model for Misuse Intrusion Detection, Proceedings of the 17th National Computer Security Conference, 1994
[19] S. Northcutt, J. Novak Network Intrusion Detection, Third Edition, 2002
[20] V. Paxson et al., Dynamic Application-Layer Protocol Analysis for Network Intrusion Detection, 15th USENIX Security Symposium 2006
[21] Y. Waizumi, H. Tsunoda, M. Tsuji, Y. Nemoto, " A Multi-Stage Network Anomaly Detection Method for Improving Efficiency and Accuracy", *Journal of Information Security,* 2012
[22] www.iss.net

Nicoleta STANCIU is an economist, PhD Candidate at the Bucharest University of Economic Studies. Her main research area is Computer Security, Information Security Management Systems, Risk Management for Information Technology Systems, IT Audit, methods and tools for implementation of Information Security Systems.