Intrusion Detection Systems (IDS) Part 2 - Classification; methods; techniques

www.windowsecurity.com/articles-tutorials/intrusion_detection/IDS-Part2-Classification-methods-techniques.html

by **Przemyslaw Kazienko & Piotr Dorosz** [Published on *15 June 2004* / Last Updated on *15 June 2004*]

Due to a growing number of intrusion events and also because the Internet and local networks have become so ubiquitous, organizations are increasingly implementing various systems that monitor IT security breaches. This is the second article devoted to these systems. The previous article dealt with IDS categorization and architecture. At this point we will provide further in depth guidance. This includes an overview of the classification of intrusion detection systems and introduces the reader to some fundamental concepts of IDS methodology: audit trail analysis and on-the-fly processing as well as anomaly detection and signature detection approaches. We will also discuss the primary intrusion detection techniques.

Classification of intrusion detection systems

Primarily, an IDS is concerned with the detection of hostile actions. This network security tool uses either of two main techniques (described in more detail below). The first one, ***anomaly detection***, explores issues in intrusion detection associated with deviations from normal system or user behavior. The second employs ***signature detection*** to discriminate between anomaly or attack patterns (signatures) and known intrusion detection signatures. Both methods have their distinct advantages and disadvantages as well as suitable application areas of intrusion detection.

When considering the area being the source of data used for intrusion detection, another classification of intrusion detection systems can be used in terms of the type of the protected system. There is a family of IDS tools that use information derived from ***a single host*** **(*system*)** — *host based* IDS (***HIDS)*** and those IDSs that exploit information obtained from a whole segment of ***a local network*** (*network based IDS, i.e.* ***NIDS****).*

Two primary types of HIDS can be distinguished [Dor02b]:

- Systems that monitor incoming connection attempts (RealSecure Agent, PortSentry). These examine host-based incoming and outgoing network connections. These are particularly related to the unauthorized connection attempts to TCP or UDP ports and can also detect incoming portscans.
- Systems that examine network traffic (packets) that attempts to access the host. These systems protect the host by intercepting suspicious packets and looking for aberrant payloads (packet inspection).
- Systems that monitor login activity onto the networking layer of their protected host (HostSentry). Their role is to monitor log-in and log-out attempts, looking for unusual activity on a system occurring at unexpected times, particular network locations or detecting multiple login attempts (particularly failed ones).
- Systems that monitor actions of a super-user (root) who has the highest privileges (LogCheck). IDS scans for unusual activity, increased super-user activity or actions performed at particular times, etc.
- Systems that monitor file system integrity (Tripwire, AIDE). Tools that have this ability (*integrity checker*) allow the detection of any changes to the files that are critical for the operating system.
- Systems that monitor the system register state (Windows platform only). They are designed to detect any illegal changes in the system register and alert the system administrator to this fact.
- Kernel based intrusion detection systems [Els00]. These are especially prevalent within Linux (LIDS, OpenWall). These systems examine the state of key operating system files and streams, preventing buffer overflow, blocking unusual interprocess communications, preventing an intruder from attacking the system. In addition, they can block a part of the actions undertaken by the super-user (restricting

privileges).

The HIDS reside on a particular computer and provide protection for a specific computer system. They are not only equipped with system monitoring facilities but also include other modules of a typical IDS, for example the response module (see Part I of the cycle).

HIDS products such as Snort, Dragon Squire, Emerald eXpert-BSM, NFR HID, Intruder Alert all perform this type of monitoring.

The network-based type of IDS (NIDS) produces data about local network usage. The NIDS reassemble and analyze all network packets that reach the network interface card operating in *promiscuous mode*. They do not only deal with packets going to a specific host – since all the machines in a network segment benefit from the protection of the NIDS. Network-based IDS can also be installed on active network elements, for example on routers.

Since intrusion detection (for example flood-type attack) employs statistical data on the network load, a certain type of dedicated NIDS can be separately distinguished, for example, those that monitor the traffic (Novell Analyzer, Microsoft Network Monitor). These capture all packets that they see on the network segment without analyzing them and just focusing on creating network traffic statistics.

Typical network-based intrusion systems are: Cisco Secure IDS (formerly NetRanger), Hogwash, Dragon, E-Trust IDS.

Certain authors (for example [Int02]) consider a blend of HIDS and NIDS as a separate class of a *Network Node IDS (NNIDS*) which has its agents deployed on every host within the network being protected (typical NIDS uses network agents to monitor whole LAN segments). In fact, a NNIDS operates very much like a hybrid per-host NIDS since a single agent usually processes the network traffic directed to the host it runs upon (an “every man for himself approach”). The main reason for introducing such hybrid IDS was the need to work online with encrypted networks and their data destined to the single host (only the source and destination can see decrypted network traffic). Most large commercially offered intrusion detection systems are shim-hybrid ones, i.e. those that merge the strengths of HIDS and NIDS in a unique concept.

The HIDS that look only at their host traffic can easily detect local-to-local attacks or local-to-root attacks, since they have a clear concept of locally available information, for example they can exploit user IDS. Also, anomaly detection tools feature a better coverage of internal problems since their detection ability is based on the normal behavior patterns of the user.

The IDS can operate as standalone, centralized applications or integrated applications that create a distributed system. The latter have a particular architecture with autonomous agents that are able to take preemptive and reactive measures and even to move over the network. The AAFID architecture of these systems has been presented in Part I of the cycle.

One may categorize intrusion detection systems in terms of behavior i.e., they may be ***passive*** (those that simply generate alerts and log network packets). They may also be ***active*** which means that they detect and respond to attacks, attempt to patch software holes before getting hacked or act proactively by logging out potential intruders, or blocking services. This is discussed in Part III of the cycle.

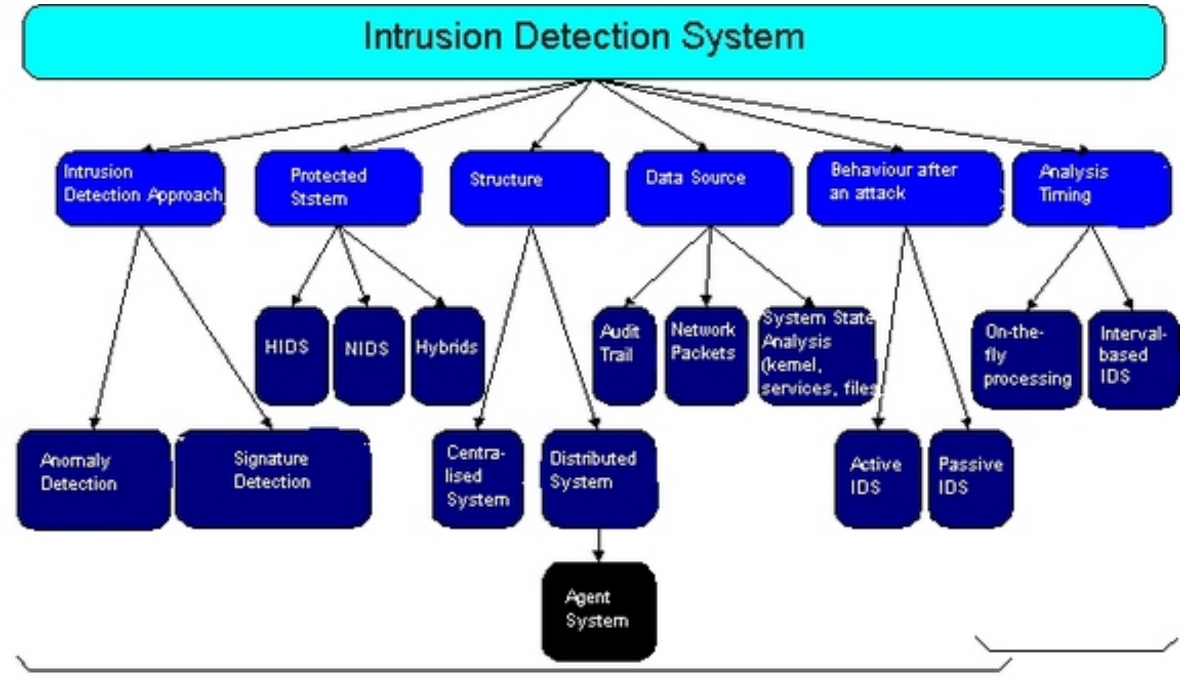

Figure 1: *Classification of intrusion detection systems*

Audit trail processing vs. on-the-fly processing

Intrusion detection systems can run on either a continuous or periodic feed of information (Real-time IDS and Interval-based IDS respectively) and hence they use two different intrusion detection approaches. ***Audit trail analysis*** is the prevalent method used by periodically operated systems. In contrast, the IDS deployable in real-time environments are designed for online monitoring and analyzing system events and user actions.

Audit Trail Processing

There are many issues related to audit trail (event log) processing. Storing audit trail reports in a single file must be avoided since intruders may use this feature to make unwanted changes. It is far better to keep a certain number of event log copies spread over the network, though it would imply adding some overheads to both the system and network.

Further, from the functionality point of view, recording every event possible means a noticeable consumption of system resources (both the local system and network involved). Log compression, instead, would increase the system load. Specifying which events are to be audited is difficult because certain types of attacks may pass undetected. It is also difficult to predict how large audit files can be – through experience one can only make a rough estimate. Also, an appropriate setting of a storage period for current audit files is not a straightforward task. In general, this depends on a specific IDS solution and its correlation engine. Certainly, archive files should be stored as copies for retrieval analysis purposes.

Log processing systems are vulnerable to Denial of Service (DoS) attacks that render audit mechanisms unreliable and unusable by overflowing the system's free space.

The main reasons for having an audit function include:

- detection of attack manifestations for post-mortem analysis;
- detection of recurring intrusion activity (yielding unauthorized privileges, abuse, attack attempts);
- identification of successful intruders;

- identification of own system weaknesses;
- Development of access and user signatures and definition of network traffic rules that are important for anomaly detection-based Intrusion Detection Systems.
- Repelling potential intruders by simply making them aware of the existence of the auditing means.
- The audit reporting may provide a form of defense for an innocent user, for example possible involved in hacking attempts.
- The log event-based IDS method needs to have the following capabilities:
- Allowing of parameterization for easy recording of system event logs and user activities,
- Providing an option of self-disengagement of logging mechanisms in the event of insufficient space or DoS attacks;
- Audit trail processing using additional mechanisms (aggregation, artificial intelligence, and data mining) because of large file sizes,
- A reasonable minimum system resource consumption for auditing purposes.

Examples of intrusion detection systems that use audit trail processing are:

- SecureView that checks logs produced by CheckPoint Firewall-1
- CMDS (*Computer Misuse Detection System*). With its built-in expert system, it analyzes all event logs to recognize abnormal user behavior.
- ACID (*Analysis Console for Intrusion* Databases) — is a PHP-based analysis engine to search and process a database of incidents generated by various security tools such as IDSs, firewalls and network traffic analyzers. ACID has a user query builder, which can analyze packets down to their payload, in order to find identical alerts among databases, which match certain criteria. It can also manage alerts and generate a variety of statistics.

On-the fly processing

With on the fly processing, an IDS performs online verification of system events. Generally, a stream of network packets is constantly monitored constantly. With this type of processing, intrusion detection uses the knowledge of current activities over the network to sense possible attack attempts (it does not look for successful attacks in the past).

Given the computation complexity, the algorithms that are used here are limited to quick and efficient procedures that are often algorithmically simple. This is due to a compromise between the main requisite – attack detection capability and the complexity of data processing mechanisms used in the detection itself.

At the same time, construction of an on-the-fly processing IDS tool requires a large amount of RAM (buffers) since no data storage is used. Therefore, such an IDS may sometime miss packets, because realistic processing of too many packets is not available.

The amount of data collected by the detector is small since it views only buffer contents. Hence, only small portions of information can be analyzed for searching certain values or sequences.

The main method used in real-time detecting is simply looking for character strings in the transport layer packets (3. and 4.), particularly in their headers. This may be done by monitoring IP addresses that initiate connections, or by checking for misappropriate TCP/IP flag combination (to capture packets that do not match known standards)[Fre01]. An example of packet pathology is when both the source and destination port addresses are set to 21. This is not compliant with FTP specifications since the source port number must be greater than 1024. Another example may be 0 Value Service-type, a packet with SYN and FIN flags both set, mismatch in numbering of sequences or acknowledgements, ACK value set to a non-zero number with the ACK flag not set, etc.

As a contrast with standard inspection methods, only selective packets in a data stream get inspected, and the inspection process only looks for "state" information, such as whether a packet contains malicious code.

A somewhat different method is applied in the application layer analysis (FTP, POP3, HTTP, etc.). Application-based IDS employ so-called standard packet inspection to analyze the TCP packet payload (headers are excluded). With this method, only selective, correlated packets in a data stream get examined and the inspection process looks for information about whether a packet matches typical packets (commands) of a given protocol. Thus, POP3 denial of service vulnerability is exploited by saturating POP3 server with multiple requests to execute a command. Here, the attack signature is developed by the number of commands sent by a given system and by establishing the alarm threshold. The method assumes that anomalies found in packet inspection, checking of packet size and threshold values are manifestations of a denial of service attack, also at the transport layer, for example *Ping of Death* attack. Another example of standard packet inspection IDSs include detecting email viruses before they get to email boxes by looking for matching email titles or attachment names. One may also search for malicious code which may compromise the system if it is attacked by, for example, buffer overflow exploits looking for signatures that monitor the user session status to disallow, for example, listing of directory structure on a FTP server before a successful user login [Dor02a). A drawback of the high layer analysis approach lies in the fact that it is time-consuming and operating environment-dependent (application layer protocols that vary from operating system to operating system).

The real-time based IDSs offer the following advantages:

- they excel at detecting attacks in progress and even responding to (blocking) them;
- the ability to cover network-inherent security holes associated with vulnerability to many types of attacks, particularly DoS, which cannot be detected using a common audit trail analysis approach—network traffic analysis is needed here;
- the system resources are less consumed than in the case of audit trail processing.

Disadvantages include:

- Source identification is accomplished based on the network address derived from the packet (not, for example, with using the network ID[1]). The source address may be spoofed, making attacks harder to trace and respond to automatically.
- That they cannot handle encrypted packets thereby not providing essential information required for intrusion detection.
- Since the analytical module uses a limited portion of source information (buffer content only), its detection capability is limited.
- A continuous scanning of network traffic reduces the network throughout the segment on which the IDS sits. This is of particular importance when an IDS tool is deployed near the firewall.

Anomaly vs. signature detection

Intrusion detection systems must be capable of distinguishing between normal (not security-critical) and abnormal user activities, to discover malicious attempts in time. However translating user behaviors (or a complete user-system session) in a consistent security-related decision is often not that simple — many behavior patterns are unpredictable and unclear (Fig. 2). In order to classify actions, intrusion detection systems take advantage of the *anomaly detection* approach, sometimes referred to as *behavior based [Deb99]* or attack signatures i.e. a descriptive material on known abnormal behavior (*signature detection*)], [Axe00, Jon00, Kum95] also called *knowledge based.*

Figure 2: *Behavior of the user in the system [Jon00]*

Normal behavior patterns — anomaly detection

Normal behavior patterns are useful in predicting both user and system behavior. Here, anomaly detectors construct profiles that represent normal usage and then use current behavior data to detect a possible mismatch between profiles and recognize possible attack attempts.

In order to match event profiles, the system is required to produce initial user profiles to train the system with regard to legitimate user behaviors. There is a problem associated with profiling: when the system is allowed to "learn" on its own, experienced intruders (or users) can train the system to the point where previously intrusive behavior becomes normal behavior. An inappropriate profile will be able to detect all possible intrusive activities. Furthermore, there is an obvious need for profile updating and system training which is a difficult and time-consuming task.

Given a set of normal behavior profiles, everything that does not match the stored profile is considered to be a suspicious action. Hence, these systems are characterized by very high detection efficiency (they are able to recognize many attacks that are new to the system), but their tendency to generate false alarms is generally a problem.

Advantages of this anomaly detection method are: possibility of detection of novel attacks as intrusions; anomalies are recognized without getting inside their causes and characteristics; less dependence of IDSs on operating environment (as compared with attack signature-based systems); ability to detect abuse of user privileges.

The biggest disadvantages of this method are:

- A substantial false alarm rate. System usage is not monitored during the profile construction and training phases. Hence, all user activities skipped during these phases will be illegitimate.
- User behaviors can vary with time, thereby requiring a constant update of the normal behavior profile database (this may imply the need to close the system from time to time and may also be associated with greater false alarm rates).
- The necessity of training the system for changing behavior makes a system immune to anomalies detected during the training phase (false negative).

Misbehavior signatures — signature detection

Systems possessing information on abnormal, unsafe behavior (attack signature-based systems) are often used in real-time intrusion detection systems (because of their low computational complexity).

The misbehavior signatures fall into two categories:

- Attack signatures – they describe action patterns that may pose a security threat. Typically, they are presented as a time-dependent relationship between series of activities that may be interlaced with neutral ones.
- Selected text strings – signatures to match text strings which look for suspicious action (for example – calling /etc/passwd).

Any action that is not clearly considered prohibited is allowed. Hence, their accuracy is very high (low number of false alarms). Typically, they do not achieve completeness and are not immune to novel attacks.

There are two main approaches associated with signature detection (already mentioned in the section describing real-time detectors):

- Verification of the pathology of lower layer packets— many types of attacks (*Ping of Death* or *TCP Stealth Scanning*) exploit flaws in IP, TCP, UDP or ICMP packets. With a very simple verification of flags set on specific packets it is possible to determine whether a packet is legitimate or not. Difficulties may be encountered with possible packet fragmentation and the need for re-assembly. Similarly, some problems may be associated with the TCP/IP layer of the system being protected. It is well known that hackers use packet fragmentation to bypass many IDS tools [Dor02a].
- Verification of application layer protocols — many types of attacks (*WinNuke*) exploit programming flaws, for example, out-of-band data sent to an established network connection. In order to effectively detect such attacks, the IDS must have implemented many application layer protocols.

The signature detection methods have the following advantages: very low false alarm rate, simple algorithms, easy creation of attack signature databases, easy implementation and typically minimal system resource usage.

Some disadvantages:

- Difficulties in updating information on new types of attacks (when maintaining the attack signature database updated as appropriate).
- They are inherently unable to detect unknown, novel attacks. A continuous update of the attack signature database for correlation is a must.
- Maintenance of an IDS is necessarily connected with analyzing and patching of security holes, which is a time-consuming process.
- The attack knowledge is operating environment–dependent, so misbehavior signature-based intrusion detection systems must be configured in strict compliance with the operating system (version, platform, applications used etc.)
- They seemed to have difficulty handling internal attacks. Typically, abuse of legitimate user privileges is not sensed by the system as a malicious activity (because of the lack of information on user privileges and attack signature structure).

Commercially offered IDS products often use the signature detection method for two reasons. Firstly, it is easier for a given signature to be associated with a known attack abstraction and to assign a name to an attack, e.g. *Ping of Death*, which is used worldwide (a suitable signature can be attached to the installation version), than the normal behavior of a certain John Brown in an organization. Secondly, the attack signature database must be updated regularly (by adding signatures of newly discovered attacks and exploits), which may create a fairly good source of income for vendors of IDS tools. A database update is at the same time a less cumbersome task than that associated with the change of typical user behavior profiles. In the latter case, a temporary closing down of the system may be required, what cannot be tolerated on certain applications.

The example below presents an attack signature taken from the Snort program that detects ping ICMP packets larger than 800 bytes, incoming from an external network and associated with any port:

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg:"MISC large ICMP"; dsize: >800; reference:arachnids,246; classtype:bad-unknown; sid:499;)

Parameter Pattern Matching

The third method of intrusion detection is subtler than the two mentioned earlier. It reasons on the fact, that system administrators monitor various systems and network attributes (not necessarily targeting security issues). As a rule, information obtained in this way has a constant specific environment. This method involves the use of

day-to-day operational experience of the administrators as the basis for detecting anomalies. It can be considered as a special case of Normal Profile Methods. The difference lies in the fact that a profile here is part of the human knowledge.

This is a very powerful technique, since it allows intrusions based on unknown type attacks. The system operator can detect subtle changes that are not obvious to the operator himself. Its inherent disadvantage is connected with the fact that humans can process and hence understand only a limited portion of information at a time, what means that certain attacks may pass undetected.

Data processing techniques used in intrusion detection systems

Depending on the type of approach taken in intrusion detection, various processing mechanisms (techniques) are employed for data that is to reach an IDS[2]. Below, several systems are described briefly:

- **Expert systems**, these work on a previously defined set of rules describing an attack. All security related events incorporated in an audit trail are translated in terms of if-then-else rules. Examples are Wisdom & Sense and ComputerWatch (developed at AT&T).
- **Signature analysis** Similarly to expert System approach, this method is based on the attack knowledge. They transform the semantic description of an attack into the appropriate audit trail format. Thus, attack signatures can be found in logs or input data streams in a straightforward way. An attack scenario can be described, for example, as a sequence of audit events that a given attack generates or patterns of searchable data that are captured in the audit trail. This method uses abstract equivalents of audit trail data. Detection is accomplished by using common text string matching mechanisms. Typically, it is a very powerful technique and as such very often employed in commercial systems (for example Stalker, Real Secure, NetRanger, Emerald eXpert-BSM).
- **Colored Petri Nets** The Colored Petri Nets approach is often used to generalize attacks from expert knowledge bases and to represent attacks graphically. Purdue University's IDIOT system uses Colored Petri Nets. With this technique, it is easy for system administrators to add new signatures to the system. However, matching a complex signature to the audit trail data may be time-consuming. The technique is not used in commercial systems.
- **State-transition analysis** Here, an attack is described with a set of goals and transitions that must be achieved by an intruder to compromise a system. Transitions are represented on state-transition diagrams.
- **Statistical analysis approach** This is a frequently used method (for example SECURENET). The user or system behavior (set of attributes) is measured by a number of variables over time. Examples of such variables are: user login, logout, number of files accessed in a period of time, usage of disk space, memory, CPU etc. The frequency of updating can vary from a few minutes to, for example, one month. The system stores mean values for each variable used for detecting exceeds that of a predefined threshold. Yet, this simple approach was unable to match a typical user behavior model. Approaches that relied on matching individual user profiles with aggregated group variables also failed to be efficient. Therefore, a more sophisticated model of user behavior has been developed using short- and long-term user profiles. These profiles are regularly updated to keep up with the changes in user behaviors. Statistical methods are often used in implementations of normal user behavior profile-based Intrusion Detection Systems.
- **Neural Networks** Neural networks use their learning algorithms to learn about the relationship between input and output vectors and to generalize them to extract new input/output relationships. With the neural network approach to intrusion detection, the main purpose is to learn the behavior of actors in the system (e.g., users, daemons). It is known that statistical methods partially equate neural networks. The advantage of using neural networks over statistics resides in having a simple way to express nonlinear relationships between variables, and in learning about relationships automatically. Experiments were carried out with neural network prediction of user behaviors. From the results it has been found that the behavior of UNIX super-users (*roots*) is predictable (because of very regular functioning of automatic

system processes). With few exceptions, behavior of most other users is also predictable. Neural networks are still a computationally intensive technique, and are not widely used in the intrusion detection community.

- **User intention identification** This technique (that to our knowledge has only been used in the SECURENET project) models normal behavior of users by the set of high-level tasks they have to perform on the system (in relation to the users' functions). These tasks are taken as series of actions, which in turn are matched to the appropriate audit data. The analyzer keeps a set of tasks that are acceptable for each user. Whenever a mismatch is encountered, an alarm is produced.
- **Computer immunology** Analogies with immunology has lead to the development of a technique that constructs a model of normal behavior of UNIX network services, rather than that of individual users. This model consists of short sequences of system calls made by the processes. Attacks that exploit flaws in the application code are very likely to take unusual execution paths. First, a set of reference audit data is collected which represents the appropriate behavior of services, then the knowledge base is added with all the known "good" sequences of system calls. These patterns are then used for continuous monitoring of system calls to check whether the sequence generated is listed in the knowledge base; if not — an alarm is generated. This technique has a potentially very low false alarm rate provided that the knowledge base is fairly complete. Its drawback is the inability to detect errors in the configuration of network services. Whenever an attacker uses legitimate actions on the system to gain unauthorized access, no alarm is generated.
- **Machine learning** This is an artificial intelligence technique that stores the user-input stream of commands in a vectorial form and is used as a reference of normal user behavior profile. Profiles are then grouped in a library of user commands having certain common characteristics [Mar01].
- **Data mining** generally refers to a set of techniques that use the process of extracting previously unknown but potentially useful data from large stores of data. Data mining method excels at processing large system logs (audit data). However they are less useful for stream analysis of network traffic. One of the fundamental data mining techniques used in intrusion detection is associated with *decision trees* [Fan01]. Decision tree models allow one to detect anomalies in large databases. Another technique refers to segmentation, allowing extraction of patterns of unknown attacks [Lee00b]. This is done by matching patterns extracted from a simple audit set with those referred to warehoused unknown attacks [Lee00a]. A typical data mining technique is associated with finding *association rules*. It allows one to extract previously unknown knowledge on new attacks [Bas00] or built on normal behavior patterns. Anomaly detection often generates false alarms. With data mining it is easy to correlate data related to alarms with mined audit data, thereby considerably reducing the rate of false alarms [Man00].

References for Part 2

[Axe00] Axelsson S.: *Intrusion Detection Systems: A Taxomomy and Survey* . Technical Report No 99-15, Dept. of Computer Engineering, Chalmers University of Technology, Sweden, March 2000, http://www.ce.chalmers.se/staff/sax/taxonomy.ps

[Bas00] Bass T.: *Intrusion Detection Systems Multisensor Data Fusion: Creating Cyberspace Situational Awareness*. Communication of the ACM, Vol. 43, Number 1, January 2000, pp. 99-105, http://www.silkroad.com/papers/acm.fusion.ids.ps.

[Deb99] Debar H., Dacier M., Wespi A.: *Towards a taxonomy of intrusion-detection systems* . Computer Networks, 31, 1999, pp. 805-822.

[Dor02a] Dorosz P., Kazienko P.: *Omijanie intrusion detection systems*. Software 2.0 no 9 (93), September 2002, pages 48-54. (In Polish only)

[Dor02b] Dorosz P., Kazienko P. *Systems wykrywania intruzów*. VI Krajowa Konferencja Zastosowan Kryptografii ENIGMA 2002, Warsaw 14-17 May 2002 , p. TIV 47-78, (In Polish only) http://www.enigma.com.pl/konferencje/vi_kkzk/index.htm

[Els00] Elson D. : *Intrusion Detection, Theory and Practice*. March 27, 2000, http://online.securityfocus.com/infocus/1203

[Fan01] Fan W., Miller M., Stolfo S., Lee W., Chan P.: *Using Artificial Anomalies to Detect Unknown and Known Network Intrusions*. In Proceedings of the First IEEE International Conference on Data Mining, San Jose, CA, November 2001, http://www.cc.gatech.edu/~wenke/papers/artificial_anomalies.ps

[Fre01] Frederick K. K.: *Network Intrusion Detection Signatures*. December 19, 2001, http://online.securityfocus.com/infocus/1524

[Int02] *Intrusion Detection Systems (IDS). Group Test (Edition 3)*, NSS Group, July 2002, http://www.nss.co.uk/ids/edition3/index.htm

[Jon00] Jones A.K., Sielken R.S.: *Computer system intrusion detection: a survey*. 09.02.2000, http://www.cs.virginia.edu/~jones/IDS-research/Documents/jones-sielken-survey-v11.pdf

[Kum95] Kumar S.: *Classification and detection of computer intrusions*. A Ph.D. Thesis. Purdue University. 1995, http://ftp.cerias.purdue.edu/pub/papers/sandeep-kumar/kumar-intdet-phddiss.pdf

[Lee00a] Lee W. i inni: *A data mining and CIDF based approach for detecting novel and distributed intrusions*. Recent Advances in Intrusion Detection, Third International Workshop, RAID 2000, Toulouse, France, October 2-4, 2000, Proceedings. Lecture Notes in Computer Science 1907 Springer, 2000, pp. 49-65. http://www.cc.gatech.edu/~wenke/papers/lee_raid_00.ps

[Lee00b] Lee W., Stolfo S, Mok K.: *Adaptive Intrusion Detection: a Data Mining Approach*. Artificial Intelligence Review, 14(6), December 2000, pp. 533-567, http://www.cc.gatech.edu/~wenke/papers/ai_review.ps

[Mar01] Marin J., Ragsdale D., Surdu J.: *A Hybrid Approach to the Profile Creation and Intrusion Detection*. Proceedings of the DARPA Information Survivability Conference and Exposition – DISCEX 2001, June 2001, http://www.itoc.usma.edu/Documents/Hybrid_DISCEX_AcceptedCopy.pdf

[Man00] Manganaris S., Christensen M., Zerkle D., Hermiz K.: *A data mining analysis of RTID alarms*. Computer Networks, 34, 2000, pp. 571-577.

[1] It is due to the lack of access to information stored in a local system (such as, for example, user name), that is usually sufficient to identify who was responsible for an incident.

[2] See particularly [Deb99].